CA Datacom®/DB

DBUTLTY Reference Guide for z/VSE

Version 12.0, First Edition

This Documentation, which includes embedded help systems and electronically distributed materials (hereinafter referred to as the “Documentation”), is for your informational purposes only and is subject to change or withdrawal by CA at any time.

This Documentation may not be copied, transferred, reproduced, disclosed, modified or duplicated, in whole or in part, without the prior written consent of CA. This Documentation is confidential and proprietary information of CA and may not be disclosed by you or used for any purpose other than as may be permitted in (i) a separate agreement between you and CA governing your use of the CA software to which the Documentation relates; or (ii) a separate confidentiality agreement between you and CA.

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

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

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

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

The manufacturer of this Documentation is CA.

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

Copyright © 2015 CA. All rights reserved. All trademarks, trade names, service marks, and logos referenced herein belong to their respective companies.

CA Technologies Product References

This document references the following CA Technologies products:

- CA Datacom®/DB
- CA Datacom® CICS Services
- CA Datacom® Datadictionary™ (Datadictionry)
- CA Datacom® DB2 Transparency
- CA Datacom® DL1 Transparency
- CA Datacom® Fast Restore
- CA Datacom® IMS/DC Services
- CA Datacom® Presspack
- CA Datacom® Server
- CA Datacom® SQL (SQL)
- CA Datacom® STAR
- CA Datacom® TOTAL Transparency
- CA Datacom® VSAM Transparency
- CA Dataquery™ for CA Datacom® (CA Dataquery)
- CA Dynam®/D Disk Management for z/VSE (CA Dynam/D for z/VSE)
- CA eMail+
- CA Ideal™ for CA Datacom® (CA Ideal)
- CA IPC
- CA Librarian®
- CA CIS
- CA Workload Automation Restart Option for z/OS Schedulers (CA WA Restart Option for z/OS Schedulers), formerly CA 11™ Workload Automation Restart and Tracking

Contact CA Technologies

Contact CA Support

For your convenience, CA Technologies provides one site where you can access the information that you need for your Home Office, Small Business, and Enterprise CA Technologies products. At http://ca.com/support, you can access the following resources:

- Online and telephone contact information for technical assistance and customer services
- Information about user communities and forums
- Product and documentation downloads
- CA Support policies and guidelines
- Other helpful resources appropriate for your product

Providing Feedback About Product Documentation

If you have comments or questions about CA Technologies product documentation, you can send a message to techpubs@ca.com.

To provide feedback about CA Technologies product documentation, complete our short customer survey which is available on the CA Support website at http://ca.com/docs.

Contents

Chapter 1: Introduction 25

CA IPC and CA Ideal Utilities 25
JCL Example Notation 25
Reading Syntax Diagrams 26
- Statement Without Parameters 27
- Statement with Required Parameters 27
- Delimiters Around Parameters 27
- Choice of Required Parameters 27
- Default Value for a Required Parameter 28
- Optional Parameter 28
- Choice of Optional Parameters 28
- Repeatable Variable Parameter 29
- Separator with Repeatable Variable and Delimiter 29
- Optional Repeatable Parameters 30
- Default Value for a Parameter 30
- Variables Representing Several Parameters 30

Chapter 2: Using CA Datacom/DB Utility (DBUTLTY) 33

Multiple Commands 33
- Prerequisite Commands 33
- Syntax Check 33
- Messages 34
- JCL Requirements 34

Command Syntax 35
- Command 35
- Keyword Parameters 35
- Line Length 35
- End of Command 36
- Comments 36

Console Commands 37
- DBUTLTY STATUS 37

DBUTLTY Dynamic Extend Considerations 39
Impact of Table Partitioning on DBUTLTY Functions 40
- The BACKUP Utility 40
- The EXTRACT Utility 41
- The LOAD Utility 41

The REORG Utility....42
Forward Recovery Utility....42
Area Level DBUTLTY Control....43
MULTUSE= Keyword....43
Database ACCESS....46
Area level ACCESS....48
COMM CLOSE Function....49
Dynamic System Table....49
Summary of Restrictions and Requirements....50
Utility Function Summary....50
DBUTLTY Sample Reports....54

Chapter 3: ACCESS (Restricting Opens or Maintenance) 55

ACCESS....55
Length of Time to Process....55
Buffer Writes....56
Area Level DBUTLTY Control....56
How to Use the Console Command....57
How to Use the DBUTLTY Command....58
Example JCL....62
Sample Report: ACCESS Function....63

Chapter 4: ACCT (Accounting Facility) 65

Activating the Accounting Facility....65
ACCT CATALOG (Catalog Accounting Database)....66
When to Use....66
PRM Table Requirements....66
Validating the Database Definition....66
DSF Modules....67
How to Use....67
Example JCL....68
Sample Report: ACCT CATALOG....68
ACCT CLOSE (Close Accounting Tables)....71
When to Use....72
Length of Time to Process....72
Multi-User Facility Connection....72
Scheduling OPENs and CLOSEs....72
How to Use the Console Command....73
How to Use the DBUTLTY Command....73
Example JCL....74
Sample Report: ACCT CLOSE....75

ACCT EDIT (Edit Accounting Database Definitions) 75
When to Use 76
Database Definitions in TEST Status 76
DSF Modules 76
How to Use 76
Example JCL 78
Sample Report: ACCT EDIT 79
ACCT OPEN (Open Accounting Tables) 81
When to Use 82
Scheduling OPENs and CLOSEs 82
How to Use the Console Command 82
How to Use the DBUTLTY Command 83
Example JCL 84
Sample Report: ACCT OPEN 84
ACCT SPILL (Force Accounting Data Recording) 85
When to Use 85
Length of Time to Process 85
How to Use the Console Command 86
How to Use the DBUTLTY Command 86
Example JCL 87
Sample Report: ACCT SPILL 88

Chapter 5: BACKUP (Create Backups) 89

BACKUP CXX (Back Up Directory) 89
When to Use 89
Backup Output 90
Databases Accessed by SQL Statements 90
How to Use 90
Example JCL (Output to Tape) 93
Example JCL (Output to Disk) 94
Sample Report: BACKUP CXX 94
Back Up Data Area 95
When to Use 95
Native Sequence Backups 96
Physical Sequence Backups (Not Continuous Operation) 96
Backups During Continuous Operation 96
Compression 96
Effects of Reorganization on Recovery 97
Backup for Forward Recovery 97
Non-DBUTLTY Backups 97
Backing Up Compressed Data Areas (Physical Sequence, UPDATE=NO) 98

Restoring a Damaged Index 98
Backup Format 98
Master List Requirements 99
Table IDs 99
Databases Accessed by SQL Statements 99
Successful Execution Requirements and Controls 99
How to Use 101
Example JCL (Output to Tape) - Native Key Sequence 110
Example JCL (Output to Disk) - Native Key Sequence 110
Sample Report: BACKUP Data Area - Native Key Sequence 111
Example JCL (Output to Tape) - Physical Sequence 112
Sample Report: BACKUP Data Area - Physical Sequence 113

Chapter 6: COMM (Communicate with Multi-User Facility) 115

Overview 115
Length of Time to Process a Request 115
COMM ALTER (Modify Error Table) 116
When to Use 116
How to Use 116
Example JCL 123
Sample Report: COMM ALTER 124
COMM CLOSE (Close Database or Area) 124
When to Use 125
How to Use the Console Command 125
How to Use the DBUTLTY Command 126
Example JCL 128
Sample Report: COMM CLOSE 129
COMM CLRPXX (Clear Statistics and Diagnostics Area (PXX)) 129
When to Use 130
Reporting PXX Data 130
How to Use the Console Command 130
How to Use the DBUTLTY Command 131
Example JCL 132
Sample Report: COMM CLRPXX 132
COMM CONSOLE (Simulate Console Command) 133
When to Use 133
How to Use the DBUTLTY Command 133
Example JCL 135
COMM EOJ (End Multi-User Facility Processing) 135
When to Use 136
Processing of EOJ Requests 136

How to Use the Console Command 137
How to Use the DBUTLTY Command 137
Example JCL 139
Sample Report: COMM EOJ 139
Summary Reports Produced 140
EOJ Reports 151
COMM NEWRXX (Request New Recovery File) 162
When to Use 162
Processing NEWRXX Requests 162
How to Use the Console Command 163
How to Use the DBUTLTY Command 163
Example JCL 165
Sample Report: COMM NEWRXX 165
COMM REQABORT (Abort a Request) 166
When to Use 166
Processing a REQABORT Request 167
How to Use the Console Command 167
How to Use the DBUTLTY Command 168
Example JCL 169
Sample Report: COMM REQABORT 170
COMM SNAP (Write Dump of Master List) 171
When to Use 171
How to Use the Console Command 171
How to Use the DBUTLTY Command 172
Example JCL 173
Sample Report: COMM SNAP 174
Master List Dump Example 175
COMM STATS (Write Current Statistics) 175
When to Use 176
How to Use the Console Command 176
How to Use the DBUTLTY Command 177
Example JCL 179
Sample Report: COMM STATS 179
COMM STATUS (Print Status of Multi-User Jobs) 180
When to Use 180
How to Use the Console Command 180
How to Use the DBUTLTY Command 181
Example JCL 183
Sample Report: COMM STATUS Utility 183
Sample COMM STATUS Online 197

Chapter 7: CONFIRM (Verify Constraint Integrity) 199

CONFIRM 199
When to Use 199
Determining Constraints In Effect 200
How to Use 200
Example JCL 202
Sample Report: CONFIRM 203

Chapter 8: CXXMAINT (Maintain Directory) 205

Overview 205
CXXMAINT ALTER CBSUSE (Trial Delete a Key from CBS Use) 206
When to Use 206
Effects on Maintenance 206
How to Use 206
Example JCL 208
Sample Report: CXXMAINT ALTER CBSUSE 208
CXXMAINT ALTER DBCS (Activating DBCS Support) 209
When to Use 209
How to Use 209
Example JCL 211
Sample Report: CXXMAINT ALTER DBCS 211
CXXMAINT ALTER DSN (Modify Data Set Name) 212
When to Use 212
How to Use 212
Example JCL 214
Sample Report: CXXMAINT ALTER DSN 215
CXXMAINT ALTER DSOP (Modify Space Reclamation Option) 215
When to Use 215
Index Area Organization 216
Synchronization with CA Datacom/DB 216
How to Use 216
Example JCL 217
Sample Report: CXXMAINT ALTER DSOP 218
CXXMAINT ALTER LANGUAGE (Activating LANGUAGE Support) 218
When to Use 219
How to Use 219
Example JCL 220
Sample Report: CXXMAINT ALTER LANGUAGE 220
CXXMAINT ALTER LINK (Disable CXX Name Edit) 221
When to Use 221
How to Use 222

Example JCL 223
Sample Report: CXXMAINT ALTER LINK 224
CXXMAINT ALTER LOGGING (Start/Stop Logging by Table) 225
When to Use 225
Active Processing 225
Impact on Recovery 225
How to Use 225
Example JCL 227
Sample Report: CXXMAINT ALTER LOGGING 227
CXXMAINT ALTER OPTION2= (Activating Other Features) 228
CXXMAINT ALTER OPTION2=datafs (DATAFS Support) 228
CXXMAINT ALTER OPTION2=datahu (High Used Mark) 231
CXXMAINT ALTER OPTION2=areaev (Data Area Extent Validation) 233
CXXMAINT ALTER OPTION2=sumode (Single User Mode) 234
CXXMAINT ALTER SINGLE (Run in a Single User Environment) 235
When to Use 236
How to Use 236
Example JCL 237
Sample Report: CXXMAINT ALTER SINGLE 237
CXXMAINT CONVERT1112 (Upgrade CXX to Current Version) 238
When to Use 238
Backup Requirements 238
Directory INIT Requirements 238
How to Use 239
Example JCL 241
Sample Report: CXXMAINT CONVERT1112 241
CXXMAINT CONVERT1211 (Downgrade CXX to Prior Version) 242
When to Use 242
Backup Requirements 242
Directory INIT Requirements 242
How to Use 243
Example JCL 245
Sample Report: CXXMAINT CONVERT1211 245
CXXMAINT DDPROD (Update Directory) 246
When to Use 246
Active Processing 247
Processing a DDPROD Request 247
How to Use 249
Example JCL 251
Sample Report: CXXMAINT DDPROD 251
CXXMAINT DELETE (Delete CXX Definitions) 253
When to Use 253

Deleting a Database 253
Deleting Tables 254
How to Use 254
Example JCL 256
Sample Report: CXXMAINT DELETE 256
Example JCL 257
Sample Report: CXXMAINT DELETE 257
CXXMAINT FORMAT0102 (Base Level Full Version 12.0 Features) 258
When to Use 258
How to Use 259
Example JCL 260
Sample Report: CXXMAINT FORMAT0102 260
CXXMAINT FORMAT0201 (Base Level Restrict to Previous Release Compatible Features) 261
When to Use 261
Considerations for Using FORMAT0201 261
How to Use 262
Example JCL 263
Sample Report: CXXMAINT FORMAT0201 263
CXXMAINT PURGE (Remove Database from Directory) 264
When to Use 264
SQL Processing 265
How to Use 265
Example JCL 266
Sample Report: CXXMAINT PURGE 266

Chapter 9: DBTEST (Execute Test/Analysis Programs) 269

DBTEST Command 1 (DBURSTR) 270
How to Use 270
DBTEST Command 2 (DBURTBL) 272
DBTEST Command 3 (DBUREND) 275
DBTEST Command 4 (DBTEST) 276
DBTEST DBTSTPR (CA Datacom/DB Test Program) 277
When to Use 277
Control Statements 277
Example DBTSTPR JCL 280
Sample Report: DBTSTPR 281
DBTEST DBIDCPR (Index Analysis Program) 283
When to Use 283
Incomplete Indexes 283
Not Available with External Security 283
How to Use 284

Example DBIDCPR JCL....284
Sample Report: DBIDCPR....285
DBTEST DBIDPPR (Index Population Analysis Program)....286
When to Use....286
DBIDPPR Output....287
How to Use....287
Example DBIDPPR JCL....288
Sample Report: DBIDPPR....288

Chapter 10: DEFRAG (Defragment Index) 293

Overview....293
Processing....294
Restrictions....294
Stopping DEFRAG....294
Security Considerations....295
DEFRAG....295
When to Use....295
How to Use....295
Example JCL....298
Sample Report: DEFRAG....298

Chapter 11: EDIT (Validate Syntax) 301

EDIT....301
When to Use....301
How to Use....301
Example JCL....302
Sample Report: EDIT....302

Chapter 12: EXTBKUP (Extract from Backup File) 305

EXTBKUP....305
When to Use....306
How to Use....306
Example JCL....309
Sample Report: EXTBKUP....310

Chapter 13: EXTEND (Extend Data or Index Areas) 315

EXTEND....315
When to Use....315
How Space Is Allocated for an EXTEND....315

Successful Execution Requirements and Controls 316
How to Use 317
Example JCL 319
Sample Report: EXTEND 319

Chapter 14: EXTRACT (Extract Data Table) 321

EXTRACT 321
When to Use 321
Extracting in Native Key Sequence 321
Extracting in Physical Key Sequence 322
EXTRACT Output 322
Master List Requirements 322
Successful Execution Requirements and Controls 322
How to Use 323
Example JCL (Output to Tape) 331
Example JCL (Output to Disk) 332
Sample Report: EXTRACT 332

Chapter 15: FILEIN (Concatenate Input) 335

FILEIN 335
When to Use 335
How to Use 335
Example JCL 338
Sample Report 339

Chapter 16: FILEOUT (Define Output Data Sets) 341

FILEOUT 341
When to Use 341
How to Use 341
Examples 343
Example JCL 344
Sample Report: FILEOUT 345

Chapter 17: FLEXPOOL (Dynamic Buffers) 347

FLEXPOOL 347
When to Use 347
How to Use the Console Command 348
How to Use the DBUTLTY Command 349
Example JCL 350

Sample Report: FLEXPOOL 350

Chapter 18: INIT CXX/FXX/IXX/LXX/Data Area 353

Restrictions 353
Multivolume INITs 353
INIT CXX (Format Directory) 353
When to Use 353
Restriction 354
CXX Name and Format 354
Identifying the Directory 354
How to Use 354
Example JCL 355
Sample Report: INIT CXX 356
INIT Data Area (Format Data Area) 357
When to Use 357
Data Set Name 357
Successful Execution Requirements and Controls 357
How to Use 358
Example JCL 361
Sample Report: INIT Data Area 361
INIT IXX (Format Index Area) 362
When to Use 363
Special Rules for Databases with Multi-Dataset Index Areas 363
Data Set Name 363
Restrictions 363
Controlling the Index Block Size 364
How to Use 364
Example JCL 367
Sample Report: INIT IXX 368
INIT LXX (Format Log Area) 369
When to Use 369
Initializing Existing Log Area 369
Log Block Size 370
Variable Log Area (LXX) Block Size 370
Restrictions and Requirements 371
How to Use 371
Example JCL for Single Log Area 373
Example JCL for Multi-Volume Log Area 374
Sample Report: INIT LXX 374
INIT FXX (Format Force Area) 375
When to Use 375

Initializing an Existing Force Area 375
Force Area Block Size 376
How to Use 376
Example JCL for Force Area 378
Sample Report: INIT FXX 378

Chapter 19: LINK (Link to the CXX) 381

LINK 381
When to Use 381
How to Use 382
Example JCL 383
Sample Report: LINK 383

Chapter 20: LOAD (Build CXX or Data Area) 385

Overview 385
LOAD CXX (Build CXX) 386
When to Use 386
Restrictions 386
SQL Processing 387
Full Directory Load 387
Directory Database Load 388
How to Use 389
Example JCL (Input from Tape) 391
Example JCL (Input from Disk) 392
Sample Report: LOAD CXX 392
LOAD (Build a Data Area) 393
When to Use 393
Loading an Existing Area 394
Loading a New Area 394
Loading a Database 394
Null Loading 395
Loading Special Rules with Database having a Multi-Dataset Index 395
Dynamic Extend during LOAD 395
Loading from a Backup 395
Reorganizing Data Areas 396
Loading to Perform RECOVERY 396
Sorting to Optimize Index Processing 396
Converting to URI 397
Loading from a Sequential File 397
VSAM Input 397
User Compression 397

Loading Tables with Integrity Constraints....398
Data Validation....398
Force Default Fields....398
Loading Multiple Tables Into a Data Area....399
Adding a Table to a Data Area....399
How to Use....401
Example JCL (Input from Tape)....416
Example JCL (Input from Disk)....417
Sample Report: LOAD Data Area....418

Chapter 21: LOCK (Lock a Data Area) 423

LOCK....423
When to Use....423
How to Use....423
Example JCL....425
Sample Report: LOCK....425

Chapter 22: MASSADD (Add Records to Table) 427

MASSADD....427
When to Use....427
Use of Pipeline....427
Log Check Points and Backout....427
Unique Keys....428
Integrity Constraints....428
Use of the Multi-User Facility....428
Tables Using DBVVRPR....428
Successful Execution Requirements and Controls....429
How to Use....430
Example JCL....433
Sample Report: MASSADD....433

Chapter 23: OLREORG (Reorganize a Database Online) 435

OLREORG....435
Restrictions....436
Security Considerations....436
When to Use....436
How to Use....437
Example JCL....440
Sample Report....440

Chapter 24: RECOVERY (Rebuild a Database) 443

Types of Recovery 443
When to Use 443
RECID= Keyword 444
Impact of Data Area Reorganization 444
RXXs without End-of-File or Trailer Records 444
Multiple File Recovery Files (RXX) 445
Backups for Use with Forward Recovery 445
Continuous Processing 446
Backward Recovery 446
Multi-User Facility Connection 446
Impact of Integrity Constraints (Forward Recovery) 446
Impact of Integrity Constraints (Backward Recovery) 447
Index Option 447
How to Use Backward Recovery 448
How to Use Forward Recovery 449
RECOVERY Command 1 (RECBASE) 450
RECOVERY Command 2 (RECJOB) 451
RECOVERY Command 3 (RECOVERY) 452
Recovery Examples 463
Example 1 463
Example 2 463
Example JCL 465
Sample Report: RECOVERY 467

Chapter 25: REMOVE (Remove Key ID) 471

REMOVE 471
When to Use 471
Changing a Key Definition Length 471
How to Use 472
Example JCL 473
Sample Report: REMOVE 473

Chapter 26: REORG (Parallel BACKUP) 475

REORG 475
When to Use 475
Parallel Backup in REORG 476
Parallel Data Reorganization 478
Parallel Index Update 481
BACKUPONLY or BACKUP phase of BACKUPLOADExecution Requirements/Controls 482

LOADONLY or LOAD phase of BACKUPLOADExecution Requirements/Controls 483
How to Use 484
Example JCL 488
Sample Report 488

Chapter 27: REPLACE (Replace All Data in a Table) 491

REPLACE 491
Restrictions 491
When to Use 492
Highlights of the Utility Side of the Processing 493
Highlights of the Multi-User Facility Processing 493
Successful Execution Requirements and Controls 493
How to Use 494
Example JCL 499
Sample Report: REPLACE 500

Chapter 28: REPORT (Generate Reports) 501

Overview 501
When to Use 501
Length of Time to Execute REPORT 501
Directory (CXX) Reports 501
When to Use 502
How to Use 502
Full Directory Report 503
Example JCL 504
Sample Report: REPORT CXX - Full 504
Sample Report: REPORT CXX - Full - Special Fields 521
Type A - Directory Space Utilization 523
Example JCL 524
Sample Report: REPORT CXX - Space Utilization 524
Type B - CXX Report 527
Type I - Impact Report 527
Example JCL 527
Sample Report: REPORT CXX - Impact Report 528
Index Area (IXX) Reports 529
When to Use 530
How to Use 530
Type A - Sequenced by Table Name 532
Example JCL 533
Sample Report: REPORT IXX - Type A 533
Type B - Sequenced by Key ID 535

Example JCL....536
Sample Report: REPORT IXX - Type B....536
Types D and E - Dump of the Index Area....538
Example JCL....538
Sample Report: REPORT IXX - Type D....539
Type G - Number of Records With Unique Key Values....541
Example JCL....541
Sample Report: REPORT IXX - Type G....542
Log Area (LXX) Report....545
When to Use....545
How to Use....545
Example JCL....546
Sample Report: REPORT LXX....546
REFGROUP Report....549
Restrictions....549
When to Use....550
How to Use....550
Example JCL (Summary REFGROUP Report TYPE=S)....553
Sample Report: REFGROUP Summary....553
Example JCL (Detail REFGROUP Report TYPE=D)....556
Sample Report: REFGROUP Detail....556
Example JCL (Detail REFGROUP Report TYPE=K)....559
Sample Report: REFGROUP Detail with Key Value Information....559
Statistics and Diagnostics Area (PXX) Report....562
When to Use....563
Information Available for Reporting....563
Length of Time to Execute....564
How to Use....564
Generating a PXX Report....568
Example JCL....568
Sample Report: REPORT PXX....568
Expression Dump....590
Sample Report....593
Reporting on the Compound Boolean Selection Facility....594
Example JCL....594
Sample Report: REPORT PXX - CBS Facility Diagnostic Report....595
Statistical Population Counting Report....606
Dumping SQL Internal Control Structures....608
TYPE=DATANE - Native Sequence Efficiency Report....608
Restrictions....609
When to Use....610
How to Use....610

Example JCL: DATANE without RECORDS= Specified 612
Sample Report: DATANE without RECORDS= Specified 612
Example JCL: DATANE with RECORDS= Specified 615
Sample Report: DATANE with RECORDS= Specified 615
Example JCL: DATANE for a Compressed Table with NUMBER=0200 Specified 618
Sample Report: DATANE for a Compressed Table with NUMBER=0200 Specified 619
TYPE=DATASP - Data Area Space Report 621
Restrictions 622
When to Use 622
How to Use 622
Example JCL: DATASP without BLOCKS= Specified 624
Sample Report: DATASP without BLOCKS= Specified 624
Example JCL: DATASP with BLOCKS= Specified 626
Sample Report: DATASP with BLOCKS= Specified 627
Recovery File (RXX) Report 629
Recovery File (RXX) Analysis Report 629
Example JCL 632
Sample Report: REPORT AREA=RXX 633
Split File Analysis Report 642
How to Use 643
Data Area Space Estimation Report (Device) 649
When to Use 649
How to Use 650
Example JCL (3380) 653
Sample Report: REPORT DEVICE (3380) 653
Example JCL (FBA) 656
Sample Report: REPORT DEVICE (FBA) 656
Type H - Utility Function History Report 659
When to Use 659
How to Use 659
Example JCL 660
Sample Report: Utility Function History 661

Chapter 29: RESET (Reset Area) 665

RESET CXX (Release Directory and Database Control) 665
When to Use 665
RESET AREA=CXX Processing 665
How to Use 666
Example JCL 668
Sample Report: RESET CXX 668
RESET LXX (Reset Log Area to Empty) 669

When to Use 669
Effect on Recovery 670
How to Use 670
Example JCL 670
Sample Report: RESET LXX 671

Chapter 30: RETIX (Reconstruct Index) 673

Rebuilding the Data Space Index or Recounting Records 673
When to Use 674
How to Use 675
Rebuilding an Index Area 677
When to Use 677
Successful Execution Requirements and Controls 677
Special Rules with Database having a Multi-Dataset Index 678
Dynamic Extend during RETIX 678
How to Use 679
Example JCL 685
Sample Report: RETIX 686
Resetting the TSN 688
When to Use 689
Multi-User Facility Connection when KEYNAME=*RTSN 689
How to Use 689

Chapter 31: RXXFIX (Recover End-Of-File) 691

RXXFIX 691
When to Use 691
RESTART Processing of a Tape Without End-of-File 691
How to Use 691
Example JCL 692
Sample Report: RXXFIX 693

Chapter 32: SECURITY RESET (Clearing Security Validation Buffer) 695

SECURITY RESET 695
When to Use 695
How to Use the Console Command 695
How to Use the DBUTLTY Command 696
Example JCL 697
Sample Report: SECURITY 698

Chapter 33: SET (Changing Error Processing) 699

SET 699
When to Use 699
How to Use 700
Example JCL 705
Sample Report: SET 705

Chapter 34: SPILL and SPILLREW (Transfer Data to RXX) 707

SPILL or SPILLREW 707
When to Use Active Recovery 707
When to Use Inactive Recovery 708
SPILL Function 708
Multi-User Facility Connection 708
How to Use 708
Example JCL 709
Sample Report: SPILL 710
SPILLREW Function 711
How to Use 711
Example JCL 712
Sample Report: SPILLREW 713

Chapter 35: SPILLOPT (Transfer Data to RXX Using MAX/MIN) 715

SPILLOPT 715
How to Use 715
Example JCL 717
Sample Report: SPILLOPT SPILL=MIN 718

Chapter 36: UNLOCK (Unlock a Data Area) 719

UNLOCK 719
When to Use 719
How to Use 719
Example JCL 721
Sample Report: UNLOCK 721

Chapter 37: VERINDEX (Verify Index) 723

VERINDEX (Verify Index) 723
When to Use 723
How to Use 723
Data to Index (DATAINDEX) 725

Example JCL 726
Sample Report: DATAINDEX without an Error 726
Sample Report: DATAINDEX with an Error 727
Index to Data (INDEXDATA) 728
Example JCL 729
Sample Report: INDEXDATA 729
Example Dump: INDOUBT Index Entries 730

Index 731

Chapter 1: Introduction

CA IPC and CA Ideal Utilities

Following is a list of CA IPC and CA Ideal utilities, with references to the location of related documentation.

Product	Utility Name	Function	Refer to:
CA IPC	SCPSUTIL	Print Subsystem Utility	*CA IPC Implementation Guide*
CA IPC	VLSUTIL	VLS Utility	*CA IPC Implementation Guide*
CA Ideal	IDBATCH	Batch Execution Utility	*CA Ideal Command Reference Guide*
CA Ideal	IDUTILTY	Library Integrity Utility	*CA Ideal Administration Guide*
CA Ideal	IDUTOTRN	Object Transport Utility	*CA Ideal Working in the Environment Guide*
CA Ideal	IDUTSTRN	Source Transport Utility	*CA Ideal Working in the Environment Guide*

JCL Example Notation

This guide uses the following JCL notation.

UPPERCASE	Identifies commands, keywords, and keyword values which must be coded exactly as shown.
symbols	Symbols, such as commas, equal signs, and slashes, must be coded exactly as shown.

Do not type the following when they appear in the JCL examples. They are provided to clarify the JCL syntax.

lowercase	Identifies a value or values that you must supply.
. . .	Indicates the omission of one or more keywords or parameters that you must code according to the specific installation at your site.

Reading Syntax Diagrams

Syntax diagrams are used to illustrate the format of statements and some basic language elements. Read syntax diagrams from left to right and top to bottom.

The following terminology, symbols, and concepts are used in syntax diagrams:

- Keywords appear in uppercase letters, for example, COMMAND or PARM. These words must be entered exactly as shown.
- Variables appear in italicized lowercase letters, for example, *variable*.
- Required keywords and variables appear on a main line.
- Optional keywords and variables appear below a main line.
- Default keywords are indicated with a single arrowhead pointing to the left.
- Double arrowheads pointing to the right indicate the beginning of a statement.
- Double arrowheads pointing to each other indicate the end of a statement.
- Single arrowheads pointing to the right indicate a portion of a statement, or that the statement continues in another diagram.
- Punctuation marks or arithmetic symbols that are shown with a keyword or variable must be entered as part of the statement or command. Punctuation marks and arithmetic symbols can include the following:

,	comma	>	greater than symbol
.	period	<-	less than symbol
(	open parenthesis	=	equal sign
)	close parenthesis	¬	not sign
+	addition	-	subtraction
*	multiplication	/	division

Statement Without Parameters

The following is a diagram of a statement without parameters:

For this statement, you must write the following:

```
COMMAND
```

Statement with Required Parameters

Required parameters appear on the same horizontal line, the main path of the diagram, as the command or statement. The parameters must be separated by one or more blanks.

The following is a diagram of a statement with required parameters:

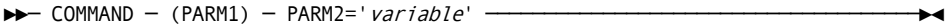
You must write the following:

```
COMMAND PARM1 PARM2
```

Delimiters Around Parameters

Delimiters, such as parentheses, around parameters or clauses must be included.

The following is a diagram of a statement with delimiters around parameters:



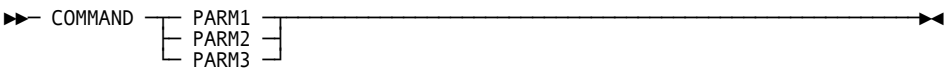
If the word *variable* is a valid entry, you must write the following:

```
COMMAND (PARM1) PARM2='variable'
```

Choice of Required Parameters

When you see a vertical list of parameters as shown in the following example, you must choose one of the parameters. This indicates that one entry is required, and only one of the displayed parameters is allowed in the statement.

The following is a diagram of a statement with a choice of required parameters:



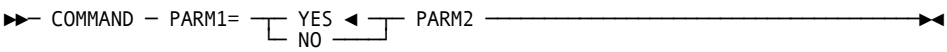
You can choose one of the parameters from the vertical list, such as in the following examples:

```
COMMAND PARM1
COMMAND PARM2
COMMAND PARM3
```

Default Value for a Required Parameter

When a required parameter in a syntax diagram has a default value, the default value appears above the main line, and it indicates the value for the parameter if the command is not specified. If you specify the command, you must code the parameter and specify one of the displayed values.

The following is a diagram of a statement with a default value for a required parameter:



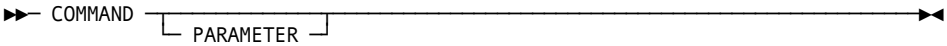
If you specify the command, you must write one of the following:

```
COMMAND PARM1=NO PARM2
COMMAND PARM1=YES PARM2
```

Optional Parameter

A single optional parameter appears below the horizontal line that marks the main path.

The following is a diagram of a statement with an optional parameter:



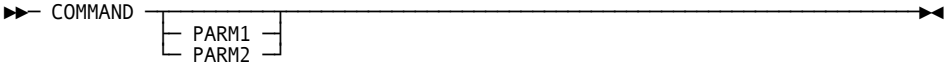
You can choose (or not) to use the optional parameter, as shown in the following examples:

```
COMMAND
COMMAND PARAMETER
```

Choice of Optional Parameters

If you have a choice of more than one optional parameter, the parameters appear in a vertical list below the main path.

The following is a diagram of a statement with a choice of optional parameters:



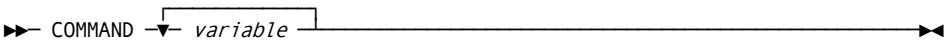
You can choose any of the parameters from the vertical list, or you can write the statement without an optional parameter, such as in the following examples:

```
COMMAND
COMMAND PARM1
COMMAND PARM2
```

Repeatable Variable Parameter

In some statements, you can specify a single parameter more than once. A repeat symbol indicates that you can specify multiple parameters.

The following is a diagram of a statement with a repeatable variable parameter:



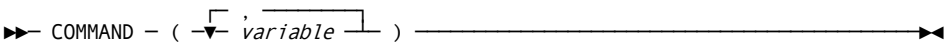
In the preceding diagram, the word *variable* is in lowercase italics, indicating that it is a value you supply, but it is also on the main path, which means that you are required to specify at least one entry. The repeat symbol indicates that you can specify a parameter more than once. Assume that you have three values named VALUEX, VALUEY, and VALUEZ for the variable. The following are some of the statements you might write:

```
COMMAND VALUEX
COMMAND VALUEX VALUEY
COMMAND VALUEX VALUEX VALUEZ
```

Separator with Repeatable Variable and Delimiter

If the repeat symbol contains punctuation such as a comma, you must separate multiple parameters with the punctuation. The following diagram includes the repeat symbol, a comma, and parentheses:

The following is a diagram of a statement with a separator with a repeatable variable and a delimiter:



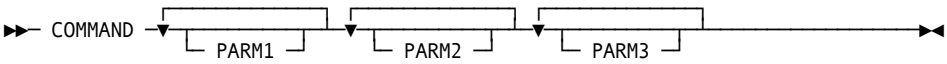
In the preceding diagram, the word *variable* is in lowercase italics, indicating that it is a value you supply. It is also on the main path, which means that you must specify at least one entry. The repeat symbol indicates that you can specify more than one variable and that you must separate the entries with commas. The parentheses indicate that the group of entries must be enclosed within parentheses. Assume that you have three values named VALUEA, VALUEB, and VALUEC for the variable.

The following are some of the statements you can write:

```
COMMAND (VALUEC)
COMMAND (VALUEB,VALUEC)
COMMAND (VALUEB,VALUEA)
COMMAND (VALUEA,VALUEB,VALUEC)
```

Optional Repeatable Parameters

The following diagram shows a list of parameters with the repeat symbol for optional repeatable parameters:



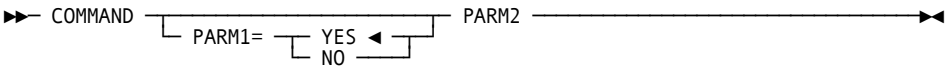
The following are some of the statements you can write:

```
COMMAND PARM1
COMMAND PARM1 PARM2 PARM3
COMMAND PARM1 PARM1 PARM3
```

Default Value for a Parameter

The placement of YES in the following diagram indicates that it is the default value for the parameter. If you do not include the parameter when you write the statement, the result is the same as if you had actually specified the parameter with the default value.

The following is a diagram of a statement with a default value for an optional parameter:

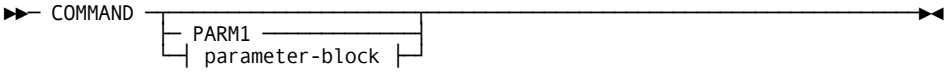


For this command, COMMAND PARM2 is the equivalent of COMMAND PARM1=YES PARM2.

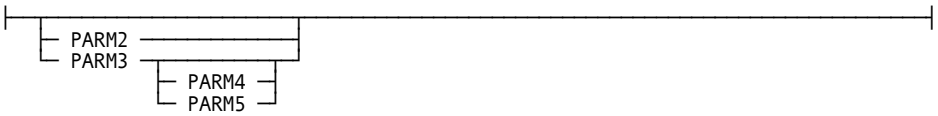
Variables Representing Several Parameters

In some syntax diagrams, a set of several parameters is represented by a single reference.

The following is a diagram of a statement with variables representing several parameters:



Expansion of parameter-block



The *parameter-block* can be displayed in a separate syntax diagram.

Choices you can make from this syntax diagram therefore include, but are not limited to, the following:

```
COMMAND PARM1
COMMAND PARM3
COMMAND PARM3 PARM4
```

Note: Before you can specify PARM4 or PARM5 in this command, youmust specify PARM3.

Chapter 2: Using CA Datacom/DB Utility (DBUTLTY)

CA Datacom/DB provides a single utility that performs all operations for creating and maintaining your information base. The program name of the utility is DBUTLTY.

This manual contains an overview of the DBUTLTY operation and details of its command syntax. The DBUTLTY functions are presented in alphabetical order. Where a console command can be used to execute the function, the syntax for the CA Datacom/DB portion of the console command is also provided.

Following the syntax information for each function is a guideline for the JCL to execute the function. Sample reports are also included with an explanation of the content of each report.

Security considerations for DBUTLTY include the following. In VM, DBUTLTY jobs must execute in the same guest machine as the Multi-User Facility. If you are using external security with CA Datacom/DB, you must have program authorization for the DBUTLTY function and the appropriate access authorization for the database resources being accessed by the DBUTLTY function. See the *CA Datacom Security Reference Guide* for more information.

Multiple Commands

You can usually perform more than one function with a single execution of DBUTLTY. Multiple commands can usually be in any order. Those functions that must be run alone are ACCT OPTION=CATALOG, ACCT OPTION=EDIT, DBTEST, and the MASSADD functions.

Prerequisite Commands

Some commands, such as DBTEST and RECOVERY, have prerequisite commands which set up the environment for them.

Syntax Check

To check the syntax of your commands, place the EDIT command first in the command stream. EDIT validates those commands that follow for syntax errors but does not execute the commands.

Messages

Every DBUTLTY execution produces a listing that contains the command exactly as entered followed by an alphabetic listing of all keywords and values entered, (except if DDOVER= is found on the statement, the data after the = (equal sign) is set to blanks to the next blank or comma). If you omitted a required keyword, that keyword prints with dashes indicating the maximum length of the keyword value and an error message. When multiple errors exist for one keyword parameter, only one error is displayed.

You also receive messages that explain any operational errors or provide explanatory information. See the *CA Datacom/DB Message Reference Guide* for a complete description of all DBUTLTY messages. Any error normally terminates execution of the utility immediately. However, the EDIT syntax check function scans all input data and displays all errors found during basic editing.

Note: Some edit errors are found in later editing and are not found with the EDIT function.

JCL Requirements

Guidelines to assist you in preparing your JCL are provided in this manual. The sample code provided in this document is intended for use as a reference aid only and no warranty of any kind is made as to completeness or correctness for your specific installation.

Samples for JCL and programs are provided in the CAIMAC install library. In z/VSE, sample PROCs allow you to make use of parameter substitution. You can copy and modify these samples for your specific requirements.

Any JOB statements should be coded to your site standards and specifications. All data set names and library names should be specified with the correct names for the installation at your site. In many examples, a REGION= or SIZE= parameter is displayed in an EXEC statement. The value displayed should be adequate in most instances, but you can adjust the value to your specific needs.

We also recommend a minimum partition size of 9 MB to handle all of the various DBUTLTY functions.

The libraries listed for searching must include the following in the order shown:

1. User libraries you may have defined.
2. CA Datacom base libraries.
3. CA IPC libraries
4. CA CIS base libraries
5. Libraries for additional CA products.

CA Dataquery users also need the following libraries and data sets for the following specific functions:

- The data set DQOUT or is used only if the DQBATCH execution uses the EXPORT function.

If you are going to use CA eMail+, include definitions of the message, folder, and network libraries in the JCL to be accessible to DQBATCH.

Command Syntax

DBUTLTY functions are invoked through commands input to the utility. Many functions also require additional information that is supplied with keywords.

Command

To execute a utility function, supply a command to DBUTLTY. Each command specifies the function to be performed. You can enter a command starting in any position of the command line, but you cannot precede a command with any other data in that line.

Keyword Parameters

A keyword parameter consists of a keyword followed by an equal sign (=) and a value. Do not code any intervening blanks when constructing a keyword parameter. Separate the command name and the first keyword parameter with at least one blank.

If you have multiple keyword parameters, separate them with commas. You can place the parameters either on one line (with no intervening blanks or other characters) or on separate lines. The keyword parameters can begin in any column. For example:

```
COMMAND KEYWORD1=VALUE1,KEYWORD2=VALUE2
```

or

```
COMMAND     KEYWORD1=VALUE1,
            KEYWORD2=VALUE2
```

Line Length

No input statement can extend beyond column 71.

End of Command

For commands without keyword parameters, the blank following the command name is the end of the command. When keyword parameters are required, DBUTLTY keeps reading input until it reads a keyword value which is followed by a blank instead of a comma.

Comments

You can include comments on your utility report by placing an asterisk (*) in column 1 of any input record. Any such input record is reproduced on your utility report but ignored for command input scan.

You can also include comments on command input records. For commands that have no keyword parameters, a comment can begin immediately following the blank that terminates the command.

For commands that have keyword parameters, you can place a comment following the last keyword parameter on a line as long as the comment is separated from the parameter (including the continuation comma) by at least one blank. A comment cannot directly follow a command that accepts keywords.

Following are examples illustrating the previous listed command input rules.

```
Input
column
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+----8

* This is a comment
* The SPILL command requires no keyword parameters,
* so the text after the command is a comment -
*
             SPILL                                Execute the spill function
*
* The LOAD command requires keyword parameters.  Comments can follow
* the last keyword parameter on a line or follow the parameter on a
* separate line.
*
             LOAD
```

```
*
* Comments cannot directly follow a command that requires keyword
* parameters such as LOAD.
*
             AREA=CXX,                          We are loading the Directory
             DTFNAME=INPUT                      From dtfname INPUT
*
* The next example shows how you might code multiple
* keyword parameters on input lines with continuations.
*
         LOAD  DBID=59,AREA=TAB,   Comments can start here
                DTFNAME=INPUT,FORMAT=BACKUP
```

Console Commands

For a comprehensive discussion about console and console-like commands, see the information about maintaining CA Datacom/DB using console commands in the *CA Datacom/DB Database and System Administration Guide*.

DBUTLTY STATUS

Console support for DBUTLTY STATUS is similar to support for Multi-User STATUS. The option is for all utilities in general, so the option has been added to the DBSYSID macro that generates the DBSIDPR loaded module. To support the console status of DBUTLTY you must therefore ensure that the DBUTLTY DBSIDPR supports console requests (DBSYSID CONSOLE=YES).

CONSOLE= provides console support for the status of CA Datacom/DB Utility (DBUTLTY) functions. CONSOLE= can be set to YES, NO, or an interval of time, with NO being the default.

Note: The option to select an interval of time is only available in z/OS environments.

If you specify YES, DBUTLTY attaches a subtask during its startup and detaches that subtask during its ending process. With CONSOLE=YES, the STATUS command is supported, meaning that you can inquire as to the status of a given DBUTLTY job by issuing a status console command. DBUTLTY functions that include status support are: BACKUP, EXTRACT, LOAD, REORG, and RETIX.

Note: The DBFLSUB callable load process is not part of DBUTLTY and does not support the console status option.

If you specify NO, you have indicated that no console support is needed.

If you specify an interval of time, CONSOLE=nn, you have indicated that the console STATUS command (CONSOLE=YES) is still supported and an automatic internal STATUS command is to be issued every nn minutes (where nn is a number from 5 to 60), the number representing minutes between intervals. As a general rule, specifying 5 minutes for the interval would be considered to be too often, while specifying 60 minutes would usually be considered to be too long.

With CONSOLE=nn selected, the first interval starts during DBUTLTY initialization. At the end of each interval, a function equivalent to the console STATUS is done. If a status is available, the message is written with the status (same message as console STATUS). If no status is available at this instant, no console message is issued. The internal STATUS does not interact with, interfere with, or hinder your ability to issue a console STATUS at any time manually.

Specifying CONSOLE=nn can be desirable for the following reasons. When long-running DBUTLTY LOAD, BACKUP, and RETIX functions are executed, you may want some way to monitor the progress of DBUTLTY. You could always manually issue a console command to request the status of a utility function that is executing, and DBUTLTY would respond with information on the number of records processed so far, the expected number of records in the function, and so on, thereby giving you an idea of the progress of the function. CONSOLE=nn, however, allows you to request that DBUTLTY *automatically* issue the console command at an interval of time determined by the value you specify with nn, instead of your having to manually issue the command. This can be particularly helpful if you are a Database Administrator, for example, at a site where your security to issue z/OS console commands is restricted.

For each DBUTLTY execution, you can select a library with the appropriate DBSIDPR assembly to indicate whether the automatic messages are desired. This information is listed to the console without your needing console authority and without having to ask an operator to issue the console command on your behalf. Specifying CONSOLE=nn has no affect on existing users. To implement it, you must do a re-assemble of DBSIDPR. Only DBUTLTY executions pointing at the new DBSIDPR are affected.

From a security viewpoint, the automatic console command feature does not display sensitive information, only information that most Database Administrators would use to monitor the progress of a long-running DBUTLTY LOAD, BACKUP, or RETIX function. DBUTLTY functionality can be secured using external security packages. Therefore, using the automatic console command feature does not cause any security exposure.

Valid Entries:

YES, NO, or nn (where nn is a number from 5 to 60)

Default Value:

NO

Communicating to a DBUTLTY Job with STATUS Command

To communicate to a DBUTLTY job:

1. Message the partition containing the DBUTLTY job.
2. At system response, enter STATUS.
3. DBUTLTY status messages are then displayed.

Messages

In support of the DBUTLTY STATUS console command, the following messages can be received:

DB01319I NO CURRENT DBUTLTY STATUS AVAILABLE

DB01320I a BASE b AREA cd ON BLOCK e OF f RECORD g h i

DB01321I a BASE b AREA c SORTING d KEYS e

DB01322I a BASE b AREA c LOADING KEY d OF e

DB01323I a BASE b AREA cd RECORD e OF ABOUT f

For more information about these messages, see the *CA Datacom/DB Message Reference Guide*.

DBUTLTY Dynamic Extend Considerations

When executing the DBUTLTY LOAD and RECOVERY functions, if the input data does not fit within the existing extents of the data area being loaded, an enhancement to the LOAD function attempts to dynamically extend the existing extents, with restrictions. Details of the dynamic extend match those that would occur in the Multi-User Facility (MUF) during ADDIT processing. Additionally, a dynamic extend is attempted when executing the DBUTLTY LOAD, RECOVERY, and RETIX functions, if the index that is needed does not fit within in the existing extents of the Index Area (IXX) or Index Areas.

In all of the cases just described, the Datadictionary attributes for dynamic extend are honored, and the DBUTLTY functions involved do not have independent keywords or options. These attributes, as during ADDIT processing, are subject to the many rules of the operating system.

The enhancements that are associated with the dynamic extend feature are automatic, if all requirements are in place, and there are no DBUTLTY keywords needing to be specified. There are restrictions, however, that if not met prevent dynamic allocation from being attempted. The restrictions specific to the DBUTLTY functions are as follows:

- For the RETIX and LOAD functions, the SORT= keyword must be specified, and with a value other than zero. Only the code path using a SORT= contains the option to dynamic extend. There is no message generated if this requirement is not met.
- Every table in the database must have been defined with the Datadictionary attribute RECOVERY YES. There is no message generated if this requirement is not met.
- For the RETIX and RECOVERY functions, the area must have been loaded with YES specified for the URI option. For the LOAD function, YES must have been specified for the URI option or allowed to default to YES for URI. There is no message generated if this requirement is not met.

For information on the DYNAMIC_EXTEND console-like command, see the *CA Datacom/DB Database and System Administration Guide*.

Impact of Table Partitioning on DBUTLTY Functions

The following topics discuss the impact of table partitioning on DBUTLTY functions.

Any Parent Table Name Restriction

The Any Parent table name cannot be used in DBUTLTY functions as a table name parameter.

Area level DBUTLTY control

In support of table partitioning, many utility functions are enhanced to support control at the area level instead of only at the full database level (see Area Level DBUTLTY Control (see page 43)).

Note: The area level controls are not available in a MUFplex.

The following utility functions have special changes because of partitioning.

The BACKUP Utility

The table name(s) provided as part of the INCLUDE or EXCLUDE keywords supports the use of a Full Parent name in addition to normal table names. The Any Parent table name is not supported. The LOAD function can load data into changed area definitions, but it is critical that all data be backed up before any is reloaded when adding, deleting, or changing the partition values of areas.

The EXTRACT Utility

The extract utility function has been enhanced to allow the Full Parent name. One specific table name is supported.

The LOAD Utility

For a LOAD with FORMAT=BACKUP, you must use the new keyword LOADID=NO to force input to output matching, using table name not table DATACOM-ID. The name must be the Full Parent name against a Full Parent or if going from ID partitioning against an unpartitioned table. If the partition criteria do not now match, the row is rejected. If a row is rejected, the LOADPTN= keyword is used to control the action taken. If it is set to SKIP, the row is ignored with no error condition. If it is set to FAIL or allowed to default to FAIL, the row is *dumped,* and the utility function terminates.

For a LOAD with FORMAT=EXTRACT, input rows that do not match the partition values for the area are treated according to the LOADPTN option as previously explained.

The LOAD utility report provides information about row movement to different areas.

The following information is provided under NOTES on the load report when tables are partitioned:

SKIPPED

Indicates record(s) from this table are to be skipped (not loaded) because they do not meet any of the range criteria for any tables in this partition.

TABLE NOT LOADED

Indicates the table was not loaded.

Database Level LOADs

Database backups and database loads are strongly suggested when going from non-partitioned to partitioned and from partitioned to non-partitioned tables.

If a database load, the data is loaded into the appropriate table based on range values.

If an area load, only data that meets the criteria for the partitioned table in that area is loaded. If a database load, data is loaded into the correct tables in an area based upon range criteria.

Area Level LOADs

If doing an area load from a database backup, the utility fails on the first record that does not fit in the area based on range values. You should code LOADPTN=SKIP if you want to continue the load and skip the records that fail to meet the criteria.

Loading from Non-Partitioned to Partitioned Tables and Back Again

The following shows the results of loading data into a table based on whether it is partitioned or not and on the parameter settings.

Non-partitioned loading to partitioned

Non-partitioned data may be loaded into a partitioned table if the non-partitioned table has the same name as the Full Parent of the partitioned table. The partition values determine to which table the row is added.

Partitioned loading to non-partitioned

Partitioned data may be loaded back to a non-partioned table if the non-partitioned table has the same name as the Full Parent of the partitioned table.

Partitioned loading to partitioned

If match on Full Parent name and then verify range values.

The REORG Utility

The REORG utility has no changes specifically for partitioning. The table name provided when using the FORMAT=EXTRACT must be a specific table and may not be a Full Parent table name.

Note: If you are partitioning by some portion of Native Key, running a REORG DTFNAME=*,INNUM=*n* (where *n* is the number of partitions) gives the current way to break up the data set.

Forward Recovery Utility

The table name optionally provided to the forward recovery may be a specific table or may be a Full Parent table. If a specific table, log records are selected with the same name or if the log record is a Full Parent name or Any Parent name and the table ID matches the specific table requested. If a Full Parent table is specified, log records are selected if the table name matches the Full Parent table, or the Any Parent table, or any specific table having the selected Full Parent name.

Area Level DBUTLTY Control

Area level DBUTLTY control gives you the option of performing a mix of utility functions on different areas of a database at the same time, by moving control from a Single User environment to a Multi-User Facility environment.

Although the majority of the DBUTLTY functions that work with data have database or area level options, prior to Version 11.0 the control was at the database level. For example, control at the database level, when initializing a data area, required the Single User utility to get full update control of the database containing the area. This meant that no other area in the database could be initialized or loaded at that time. With area level DBUTLTY control, certain DBUTLTY functions now support a MULTUSE=YES option that moves the controls from a Single User to a Multi-User Facility environment and, in addition to the full database, apply to the area level.

This feature requires that the database be optimized in the Multi-User Facility, which is done by *not* having the ACCESS Multi-User startup option specified as NOOPT for this database.

MULTUSE= Keyword

The MULTUSE= keyword, for area level DBUTLTY control, is an option in the following DBUTLTY functions:

- BACKUP for Data Areas
- EXTEND for Data Areas
- EXTRACT
- INIT for Data Areas
- LOAD for Data Areas
- Forward RECOVERY
- REORG
- REPLACE
- RETIX

The following gives details about using MULTUSE= in the previously listed functions:

- EXTEND and INIT for Data Areas

  Specifying MULTUSE=YES in the EXTEND and INIT functions for data areas allows a single area in a database to be initialized or extended with no disabling of the other areas in the database. When MULTUSE=YES, the functions execute in a new format that is a combination of Multi-User Facility and Single User modes. The database involved must be able to be opened for update in the Multi-User Facility address space. The area, however, must have no tables open. Once begun, the functions do not allow other tasks to open any tables in the area being processed. Other tables in the database, but not in the area, may be open for read or update in the same Multi-User Facility. The actual data set open is performed in the utility region, and the JCL requirements are applied to the utility region.

  When you specify or allow the default of MULTUSE=NO in the INIT or EXTEND functions for data areas, the functions execute as in versions prior to Version 11.0, that is, completely Single User in mode.

- LOAD and RETIX

  In LOAD and RETIX functions that do not use the KEYNAME= keyword, specifying MULTUSE=YES allows a single area in a database to be loaded or have its index rebuilt, with no loss of read and update access to other areas in the database. When MULTUSE=YES, the functions execute in a new format that is a combination of Multi-User Facility and Single User modes. The database involved must be able to be opened for update in the Multi-User Facility address space. The area, however, must have no tables open. If the database is currently open, CA Datacom/DB closes this area with the area close (you would need to close the area yourself *only if* the database is open and you need the Multi-User Facility to close it, so that any type of open data set lock can be necessary in order to scratch and reallocate the data set). Once started, the functions do not allow other tasks to open any tables in the area being processed. Other tables in the involved database, but not in the area, may be open for read or update in the same Multi-User Facility.

  With MULTUSE=YES:

  - The area is first opened in Multi-User Facility for the LOAD or RETIX.
  - The area is also opened in Single User mode.
  - The data area processing occurs in the utility address space.

- The key build/sort occurs in the utility address space.
- The index load/merge occurs in the Multi-User Facility address space. Packages of index entries (about 32k bytes) are sent from the utility to the Multi-User Facility.

MULTUSE=YES affects several other options. The SORT option becomes required and is an error to be specified as 0. The AREA= keyword becomes required.

When MULTUSE=NO is specified or is allowed to default in the LOAD and RETIX functions, the functions execute as in versions prior to Version 11.0, that is, the database involved may not be open for update, and all processing executes in the Single User mode.

- RECOVERY (forward only)

  Beginning with Version 11.0, using MULTUSE=YES in the RECOVERY OPTION=FORWARD function of DBUTLTY allows a single area in a database to be recovered with no disabling of the other areas in the database. Forward recovery functionality that existed in versions prior to Version 11.0 remains unchanged.

  For area level DBUTLTY control, three keywords must be provided, MULTUSE=YES, DBID=, and AREA=. The three are a set, that is, if one of the three is specified, the other two must also be specified. When all are provided, the RECBASE and RECJOB functions are not allowed. Using the three area level control keywords, recovery works at the same level as the LOAD, RETIX, INIT, EXTEND, and REORG utility functions.

  When MULTUSE=YES is specified, the area involved must be able to be opened for update in the Multi-User Facility. Once begun, the function prevents other opens for the area. Tables in other areas may be open for other utilities and user applications based upon normal rules. This recovery does not add log recordsIf it fails, it is necessary to restore the data area again and restart the recovery, the same as would be required when running any of the other forms of forward recovery.

Following are the rules for execution for forward recovery with MULTUSE=YES:

- Database must be open for update in the Multi-User Facility or able to be opened.
- No table in the area may have an open for update user.
- No table in the area should have a read user. CA Datacom/DB does, however, allow it.
- Area may not currently be being processed by another MULTUSE=YES forward recovery, INIT, EXTEND, LOAD, RETIX, or REORG.
- The area should be set with ACCESS STATUS=UTLTY. CA Datacom/DB does not, however, absolutely require it.

In forward recovery, if you specify MULTUSE=NO, or allow the function to default to it, the database involved must be able to be opened for update by the utility address space. This means it is not open for update by any other address space, including a Multi-User Facility. Also, the database should not be open for read, even though CA Datacom/DB does allow it.

- REORG

  Area level DBUTLTY control allows a single area in a database to be reorganized with no disabling of the other areas in the database.

  With MULTUSE=YES, the data-spaces containing index information are defined with a global scope, making them available to the Multi-User Facility. When MULTUSE=YES, the index phase must occur in Multi-User Facility with full interaction with other tasks processing the index and sharing the same buffer pool. Each task does single block reads. The area being reorganized cannot have other users for REORG to start and, once started, it does not allow other users to open.

- REPLACE

  The REPLACE function has table level control, but it interacts with area level controls. For example, if replacing a single table in an area, the area cannot be used by the LOAD function.

Database ACCESS

Database ACCESS controls all access (User Requirements Table open) to a database in a Multi-User Facility, with the exception of the RESTART process done during the Multi-User Facility enable.

ACCESS STATUS=WRITE

Allows all types of access.

ACCESS STATUS=OFF

Denies all types of access.

ACCESS STATUS=READ

Allows User Requirements Table opens, with UPDATE=NO specified or defaulted.

ACCESS STATUS=UTLTY

Allows all DBUTLTY functions and denies all OPENS that are not DBUTLTY, except the DBUTLTY function DBTEST with PGM=DBTSTPR, as it executes as though it is not part of DBUTLTY.

Database ACCESS is set by 3 processes:

- Multi-User startup options,
- A console command, and
- The DBUTLTY function ACCESS.

A temporary, internal, parallel form of ACCESS=OFF is done during a CATALOG from CA Datacom/DB.

The database level of ACCESS is usually done to stop, in an orderly fashion, access in a Multi-User Facility because:

- It has the ability to wait for the completion of active jobs, and
- It usually precedes both Single User forms of DBUTLTY functions or other processes, such as volume copies or restores.

  Note: DBUTLTY Single User functions do not know about the ACCESS status of any Multi-User Facility, nor are they dependent upon any current status.

Following is an example of database ACCESS used to reorganize an entire database. In the example, the DBUTLTY functions BACKUP, INIT, and LOAD each execute in Single User mode with no knowledge of any Multi-User Facility or its ACCESS status, and therefore do not require them. However, ACCESS ensures correct execution of the utility set because it ensures that no other job, which could either update data that would be lost or cause the INIT to fail with a return code 46, can open the base between the BACKUP and INIT.

1. Use ACCESS to stop new jobs and wait on current jobs:

   ```
   ACCESS STATUS=OFF,DBID=1,USERS=WAIT
   ```

2. Use COMM CLOSE to close the database:

   ```
   COMM OPTION=CLOSE,DBID=1
   ```

3. Use BACKUP to back up the entire database:

   ```
   BACKUP DBID=1,...
   ```

4. Use INIT to initialize the Index Area (IXX):

   ```
   INIT AREA=IXX,DBID=1
   ```

5. Use LOAD to load the reorganized data:

   ```
   LOAD DBID=1,FORMAT=BACKUP,...
   ```

6. Use ACCESS to allow new user jobs:

   ```
   ACCESS STATUS=WRITE,DBID=1
   ```

Area level ACCESS

Area ACCESS controls all access (User Requirements Table open) to a data area (and the table(s) the area contains) in a Multi-User Facility, with the exception of the RESTART process done during the Multi-User Facility enable.

ACCESS STATUS=WRITE

Allows all types of access.

ACCESS STATUS=OFF

Denies all types of access.

ACCESS STATUS=READ

Allows User Requirements Table opens, with UPDATE=NO specified or defaulted.

ACCESS STATUS=UTLTY

Allows all DBUTLTY functions and denies all OPENS that are not DBUTLTY, except that the DBUTLTY function DBTEST with PGM=DBTSTPR, as it executes as though it is not part of DBUTLTY.

Area level ACCESS is set by the DBUTLTY function ACCESS, controlled by users. The area level of ACCESS is usually done to stop, in an orderly fashion, access in to a Multi-User Facility area. It usually precedes a DBUTLTY function at the data area level that needs to execute in the Multi-User Facility.

A temporary, internal, parallel form of the ACCESS STATUS=UTLTY function is done by other CA Datacom/DB functions such as LOAD with MULTUSE=YES. The LOAD, in this case, validates that its requirements are met then sets the area in the protected status, thereby preventing other opens to any table in the area for the duration of the LOAD. The LOAD with MULTUSE=YES does not require the area level ACCESS.

Following is an example of an area ACCESS to reorganize an area in the Multi-User Facility with no loss of access to other areas in the Multi-User Facility. In the example, the utility functions BACKUP and LOAD each execute in the Multi-User Facility. The ACCESS function is not required, but if used it ensures correct execution of the utility set by ensuring that no other job, which could either update data that would be lost or cause the LOAD to fail with a return code 88, can open the base between the BACKUP and LOAD

1. Use ACCESS to stop new jobs and wait on current jobs:

   ```
   ACCESS STATUS=UTLTY,DBID=1,AREA=DEM,USERS=WAIT
   ```

2. Use BACKUP to back up the DEM area:

   ```
   BACKUP DBID=1,AREA=DEM,MULTUSE=YES,...
   ```

3. Use LOAD to load the reorganized data:

   ```
   LOAD DBID=1,FORMAT=BACKUP,AREA=DEM,MULTUSE=YES,...
   ```

4. Use ACCESS to allow new user jobs:

   ```
   ACCESS STATUS=WRITE,DBID=1,AREA=DEM
   ```

If there is an attempt to open (a user application open or a utility open) a table in an area that is contrary to a user-executed DBUTLTY ACCESS with the AREA keyword, return code 88(002) is generated. If there is an attempt to open a table in an area that is contrary to a DBUTLTY execution in the Multi-User Facility, return code 88(003) is generated. For example, an 88(003) would be received if a data area was being initialized using the MULTUSE=YES keyword, and there was an attempt by a user application or a utility to open a table in the area.

COMM CLOSE Function

The COMM CLOSE function of DBUTLTY either closes all data sets in a database and closes the definitions in the Directory (CXX), or it closes a single area of an open database.

Specifying the AREA=*name* keyword requires the DBID= keyword to also be specified. You would need to use this feature only if the Multi-User Facility has the area open and you want to initialize, extend, or load it through DBUTLTY and need to uncatalog it.

The CLOSE console command has not been changed.

Dynamic System Table

The MUF_ACCESS_AREA (MFN) Dynamic System Table contains information about area level DBUTLTY access. See the MUF_ACCESS_AREA information in Dynamic System Table documentation for details.

Summary of Restrictions and Requirements

User applications and CA Datacom utility functions provide execution requirements by defining a User Requirements Table and issuing an open for it. If all requirements are not met, the open is rejected. If all are met, control is established to allow the successful completion of the application or utility function.

The following restrictions related to area level DBUTLTY control and the ACCESS Multi-User startup option should be noted:

- To specify YES for the MULTUSE= keyword, in the DBUTLTY functions that have MULTUSE= as an option, the database must be running in the OPTIMIZE mode, set with the ACCESS Multi-User startup option established during the Multi-User Facility startup.
- Area level DBUTLTY control cannot be used when executing against a Multi-User Facility that is running as part of a MUFplex, when more than one Multi-User Facility is enabled. Return code 94(116) occurs when attempting to run in this configuration.
- Area level DBUTLTY control cannot be used in a database defined to be VIRTUAL. The catalog from CA Datacom/DB is still at the database level only.

Note: Area level DBUTLTY control supports, but works independently of, table partitioning.

Area level DBUTLTY control is best used when it is important for one area in a database to be open for normal processing while another area needs to be reorganized or rebuilt, presuming the applications use the *available* areas, not the unavailable area. However, if applications open tables in all areas, for example in CA Datacom/DB when one-table areas are used, we recommend that execution be allowed to run in full Single User mode, which is the default.

Utility Function Summary

You can perform the following functions with DBUTLTY. Each of these functions is documented in detail in the following chapters. The functions are discussed in alphabetical order.

Functions that have a Y in the SU column can be executed in Single User mode as well as in Multi-User mode. Any function that does not have a Y in the SU column can therefore only be executed when the MUF is enabled. An asterisk (*) in the SU column indicates one or more exceptions that are explained in the Description column.

Note: When external security is turned on, most DBUTLTY functions require the MUF to be enabled, even those running in Single User mode, because the MUF is used for authorization purposes.

Function	SU	Description
ACCESS		Allows the ability to restrict access to a table, either from the database level or the area level.
ACCT		Provides options to edit an accounting database definition, catalog the definition to the Accounting Facility, and request an open, close, or spill of individual accounting tables.
BACKUP	Y	Outputs a sequential data set containing a backup of the area specified. The Directory (CXX), any specific data area, or all areas in a database can be backed up.
COMM		Communicates a specific request to the Multi-User Facility. Only the STATUS option can be executed in both Single User mode and Multi-User mode.
CONFIRM		Verifies that the data in a specific table adheres to SQL-defined integrity constraints.
CXXMAINT		Provides functions to maintain the Directory (CXX).
DBTEST		Executes one of the following test or analysis programs: **DBTSTPR** CA Datacom/DB Test Program. The DBTSTPR program can run in Single User mode if NO is specified for the MULTUSE= keyword. **DBIDCPR** Index Area Analysis Program **DBIDPPR** Index Area Population Analysis Program
DEFRAG		Defragments or compresses the DXX, that is, the "lowest" level of the IXX that contains key values and pointers to actual rows.
EDIT	Y	Validates the syntax and content of the DBUTLTY commands and associated keywords, without invoking the functions.
EXTBKUP		Reads a backup file and writes extract data to a sequential output data set(s).
EXTEND		Formats additional space to extend an Index Area (IXX) or data area.
EXTRACT	Y	Outputs a CA Datacom/DB table in expanded form to a sequential data set.

Function	SU	Description
FILEIN		Allows you do one load with data from several backups.
FILEOUT		Used to define output data sets, for example to define one or more output data sets used by the EXTBKUP function.
FLEXPOOL		Adds or deletes dynamic data area or Index Area (IXX or DXX) buffers to the system.
INIT	Y	Formats the Directory (CXX), Log Area (LXX), Index Area (IXX), Force Area (FXX), or data area for use by CA Datacom/DB.
LINK		Links a data area or Index Area (IXX) to the Directory (CXX).
LOAD	Y	Reads data from a sequential or VSAM data set and loads this data into the specified area. LOAD only applies to a data area or the Directory (CXX).
LOCK		Locks a data area when you request a backup while the Multi-User Facility is updating the data area.
MASSADD		Reads a sequential or VSAM input data set and adds the records to a specified CA Datacom/DB table. MASSADD can run in Single User mode if NO is specified for the MULTUSE= keyword.
OLREORG		Reorganizes the database table without having to take it offline.
RECOVERY		Reprocesses maintenance transactions from a Recovery File (RXX) in either forward or backward mode. Forward recovery can be run in a Multi-User mode or in a Single User mode while the MUF is up, providing that the database is closed and not being used. Backward recovery must, however, be run in a Multi-User mode.
REMOVE		Deletes a key ID from the Index.
REORG		The parallel REORG function of the CA Datacom/DB Utility (DBUTLTY) provides a way for you to more quickly back up a large data area. By executing parallel tasks, the REORG function completes faster than a Native backup.
REPLACE		Provides the ability to load one table in a multiple table area with new data, replacing all current data for that table. It executes in the Multi-User Facility and allows full read and update access to every other table in the same area and database to other applications in the same Multi-User Facility.

Function	SU	Description
REPORT		Prints one of the following reports: ■ Directory (CXX) Report ■ Summary Space Utilization Report ■ Index Area (IXX) report ■ Database History Report ■ Log Area (LXX) Report ■ Statistics and Diagnostics Area (PXX) Report ■ Recovery File (RXX) Analysis Report ■ Split File Analysis Report ■ Data Area Space Estimation Report ■ Utility Function History Report (TYPE=H)
RESET	Y	Resets the Log Area (LXX) as empty, or resets the Directory (CXX) exclusive control flags and sets database update and read controls to zero.
RETIX		Reads the area indicated (or all areas in a database) and rebuilds the data space index or the entire Index Area (IXX) information for that area based on the current data area information. You can also use the RETIX function to reset the Transaction Sequence Number (TSN).
RXXFIX	Y	Reconstructs log records being spilled to the Recovery File (RXX), if they had no End-of-File or trailer records as a result of an operating system or Multi-User Facility abend.
SECURITY		Clears cached permissions after making security changes.
SET		Allows you to establish how you want DBUTLTY to support error handling of functions within a specific execution. SET also allows debugging tools and tuning options to be specified.
SPILL		Moves transactions that can be spilled from the Log Area (LXX). We recommend you use the SPILLOPT function instead of the SPILL function (see the following). SPILL works the same as SPILLOPT SPILL=MAX.
SPILLOPT		Allows you to specify MAX or MIN when requesting a spill. Also allows the setting to allow update of a SPILL history table to record SPILL activity.
UNLOCK		Unlocks the data area you locked with the LOCK function and allows update access to the area.

DBUTLTY Sample Reports

This manual contains many Sample Report sections to show you what to expect from the utility and to explain the entries on the reports. The headings on the reports are in one of the following formats:

```
Date: mm/dd/ccyy ❶ ********************************************************************************        Page:     1      ❷
                   *                                CA Datacom/DB                                 *
Time: hh.mm.ss   ❶ *                              General Utility                                 *      Release: 12
                   *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn       ❸
                   ********************************************************************************
```

❶

The date and time printed on the report is when the DBUTLTY assembled the report after completing the requested function.

❷

The page number and the version of CA Datacom/DB are noted.

❸

Service pack information starts at GA with SP0.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX
```

The Directory (CXX) in use is noted when appropriate.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX
```

The DATACOM-ID (also known as the DBID) of the database in use is noted when appropriate.

Chapter 3: ACCESS (Restricting Opens or Maintenance)

The ACCESS function provides the ability to allow or disallow an open of a User Requirements Table specified at the database level while the Multi-User Facility is active. You can also use it to allow or disallow the ability of applications to execute maintenance to a database while Multi-User Facility is active.

Using ACCESS against a system database is not recommended. The use of ACCESS OFF/UTLTY/READ/NOMAINT is honored for all bases except the Compound Boolean Selection (CBS) temporary index. It prevents system functions from being processed, based upon which ACCESS status, which database, and what is executing.

ACCESS

The Access function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

Length of Time to Process

When the function is processed with the keywords READ, OFF, UTLTY, or NOMAINT the database is disabled instantly. After the database or databases are disabled, a console message is written.

Buffer Writes

To provide a way to maintain better continuous operation backup, when a new access status has been processed for a database, a write of all pending buffers for the Index Area(s) and data area(s) is immediately done, unless the WAIT option is used. Such buffer writes are also done for the WRITE status so a write pending can occur with no loss of any current or planned activity.

Note: If the Multi-User Facility is restarted, all access statuses you have set with the ACCESS function are cleared. You must repeat any restrictions for access by submitting the function again.

ACCESS OFF/READ/WRITE/UTLTY controls the opening of a User Requirements Table (URT). It has no effect on an already open URT.

ACCESS MAINT/NOMAINT controls the maintenance commands to add, update, or delete rows. It has no effect on the open or close of a URT. NOMAINT instantly causes all new maintenance commands to receive an error. It does **not** affect transaction backout (ROLLBACK) or recovery.

Area Level DBUTLTY Control

Beginning with Version 11.0, a new set of WRITE/READ/OFF/UTLTY status conditions allow, independently, the control of all non-DBUTLTY opens of specific data areas. Area level access stops all Multi-User Facility applications to an area, so that the area can be processed in the Multi-User Facility itself, for example to REORG an area without loss of access to other areas in the database.

Area level access is independent of database level control, the functions and meanings of database level control.

An AREA= keyword provides the choice of data area level control. When AREA= is specified, DBID= must also be specified. For example, if data area A01 contains tables POL, POH, PNC, and PNM, the ACCESS utility with keywords STATUS=UTLTY,DBID=1,AREA=A01 prevents future opens of tables POL, POH, PNC, and PNM by every application in Multi-User Facility that is not done through DBUTLTY.

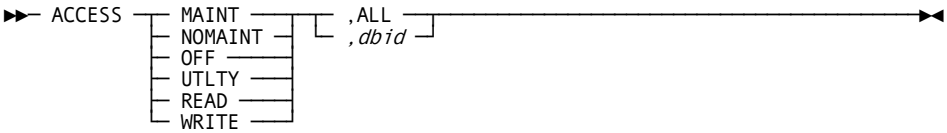
Note: The ACCESS Multi-User startup option has not been changed for support of area level DBUTLTY control, nor has the ACCESS console command been changed. Area level access is provided *only* through DBUTLTY.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

ACCESS

Invokes the function to allow or disallow an open of a User Requirements Table specified at the database level or invokes the function to allow or disallow maintenance commands while the Multi-User Facility is active.

Required Keywords

MAINT

This access level allows maintenance commands to process against the database(s).

NOMAINT

This access level blocks application maintenance commands from processing against the database(s).

OFF

This access level blocks all User Requirements Tables from opening this database(s).

UTLTY

This access level allows update access to this database by DBUTLTY and denies access to all applications that are not DBUTLTY.

Note: An exception is the DBTEST function of DBUTLTY which executes *not* with DBUTLTY authority.

READ

This access level allows read-only User Requirements Tables to open the database(s). (The User Requirements Tables can specify UPDATE=NO only.)

WRITE

This access level allows all User Requirements Tables to open the database(s). (The User Requirements Tables can specify UPDATE=YES or NO.)

Note: The preceding status conditions exist after processing this command. A previous status is ignored.

,ALL

Indicates that all databases are to participate in the ACCESS function.

,dbid

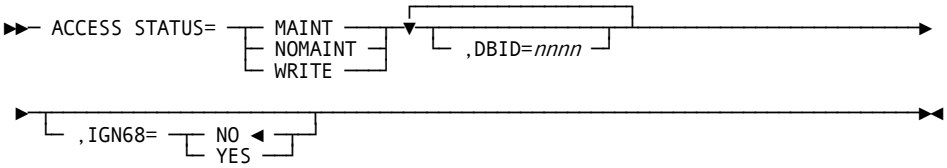
Specifies the database ID for a single database. You can specify only one per function.

Consult your site operating system manual for additional syntax information related to your operating system.

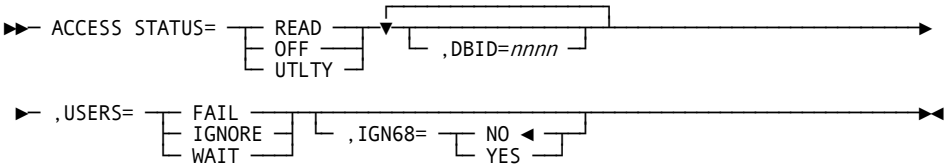
How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. To use the ACCESS function, execute DBUTLTY using the following:

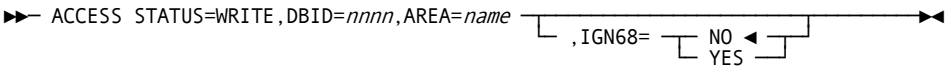
ACCESS STATUS (MAINT/NOMAINT/WRITE, Database Level)



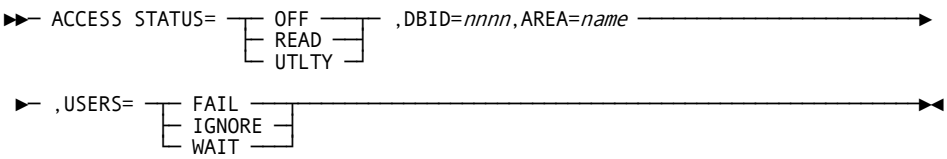
ACCESS STATUS (READ/OFF/UTLTY, Database Level)



ACCESS STATUS (WRITE, Area Level)



ACCESS STATUS (OFF/READ/UTLTY, Area Level)



Command

ACCESS

Invokes the function to specify the type of access.

Required Keywords

AREA=*name*

Specifies data area level access. If AREA= is specified, DBID= must also be specified.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

Valid area name

Default Value:

(No default)

,DBID=*nnnn*

At the area level, you must specify one database ID.

Valid Entries:

One valid database ID.

Default Value:

(No default)

STATUS=

Specifies the currently desired access level for the area.

MAINT

This access level allows maintenance commands to process against the database(s).

NOMAINT

This access level blocks application maintenance commands from processing against the database(s).

OFF

This access level blocks all User Requirements Tables from opening this area. That is, databases and areas are not allowed to be opened by any application, including DBUTLTY.

READ

This access level allows read-only User Requirements Tables to open a database or area. That is, READ allows a database or area to be opened without update by all applications, including DBUTLTY. This allows support for the DBUTLTY BACKUP and EXTRACT functions. (The User Requirements Tables can specify UPDATE=NO only.)

WRITE

This access level allows all User Requirements Tables to open the database or area. That is, WRITE allows the database or area to be opened with or without update by all applications, including DBUTLTY. (The User Requirements Tables can specify UPDATE=YES or NO.)

UTLTY

This access level allows the database or area to be opened *only* by DBUTLTY.

DBUTLTY functions that do not open a database or area are never restricted by the ACCESS function.

The DBUTLTY function DBTEST with keyword PGM=DBTSTPR is an exception. It executes as a user application without special DBUTLTY authority.

Valid Entries:

MAINT, NOMAINT, OFF, READ, WRITE, or UTLTY

Default Value:

(No default)

,USERS=

(Only required for OFF, UTLTY, or READ.) Specifies how the function is to execute. The parameter is required for STATUS=READ, UTLTY, or OFF, and it is optional, but ignored, for STATUS=WRITE. Regardless of which option is selected, the area or access level of the DBID is changed.

FAIL

Specifies that if an open User Requirements Table exists for a table within the area currently open for read or update (when OFF), or update (when READ), or not a DBUTLTY (when UTLTY), the utility step is to terminate causing no additional utility functions to execute. The Multi-User Facility can have the area open *without* having any open User Requirements Tables when using the OPTIMIZE option of the ACCESS Multi-User startup option.

IGNORE

Indicates that the ACCESS is to continue without regard to the open users.

WAIT

Specifies that if an open User Requirements Table exists for a table within the area currently open for read or update (when being changed to OFF), or update (when being changed to READ), or not a DBUTLTY (when being changed to UTLTY), the utility step is to wait, stopping additional utility functions from executing. The Multi-User Facility can have the area open *without* having any open User Requirements Tables when using the OPTIMIZE option of the ACCESS Multi-User startup option. This wait has no time limit.

Valid Entries:

FAIL, IGNORE, or WAIT

Default Value:

(No default)

Optional Keywords

,DBID=*nnnn*

At the database level, DBID= is optional and may repeat. Each occurrence may specify one or more valid database IDs or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250).

If DBID= is not specified, all databases are processed.

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database in a range not defined is ignored.

Valid Entries:

One or more valid database IDs or a range of DBIDs as already described.

Default Value:

ALL

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled. For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Example JCL

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
// EXEC DBUTLTY,SIZE=750K
          ACCESS    STATUS=READ,USERS=IGNORE
/*
/&
* $$ EOJ
```

Sample Report: ACCESS Function

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    ACCESS STATUS=READ,USERS=IGNORE

                    FUNCTION=ACCESS
                    STATUS=READ
                       USERS=IGNORE
```

The report shows the following:

- The command exactly as entered.
- An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.
- Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                     DB00607I - BASE ALL   ACCESS READ
```

The report shows the following:

- A message indicating that DBUTLTY successfully placed the request on the Multi-User Facility communication queue.

Note: This is only one example of the text that can be associated with the DB00607I message. See the *CA Datacom/DB Message Reference Guide* for a description of the possible variations.

Chapter 4: ACCT (Accounting Facility)

Use the ACCT function to edit and catalog Accounting Facility database definitions or to communicate open, close, or spill requests to the Accounting Facility running in a Multi-User Facility.

Activating the Accounting Facility

To activate the Accounting Facility, perform the following steps:

1. Define the accounting database to CA Datacom/DB. (The CA Datacom/DB User Requirements Table must be defined to use the Multi-User Facility.) See the *CA Datacom/DB Database and System Administration Guide* for details.
2. Use the ACCT EDIT option to verify that the definition is accepted by the Accounting Facility. See ACCT EDIT (Edit Accounting Database Definitions) (see page 75) for details.
3. CATALOG the accounting database. See the *CA Datacom Datadictionary User Guide* for details.
4. Use the INIT function of DBUTLTY to initialize the accounting database's data and Index Areas. See INIT Data Area (Format Data Area) (see page 357). and INIT IXX (Format Index Area) (see page 362) for details.
5. Use the LOAD function of DBUTLTY to null load each of the data areas in the accounting database. See LOAD (Build a Data Area) (see page 393) for details.
6. Change your Multi-User startup options to include the necessary Accounting Facility parameters. See the *CA Datacom/DB Database and System Administration Guide* for details.
7. Use the ACCT CATALOG option to build the accounting database definitions in the PRM table. See ACCT CATALOG (Catalog Accounting Database) (see page 66) for details.
8. Reinitialize the Multi-User Facility to activate accounting. See the *CA Datacom/DB Database and System Administration Guide* for details.

The following sections describe each of the operations that you can perform with the ACCT function.

ACCT CATALOG (Catalog Accounting Database)

The Accounting Facility uses the resources of the Multi-User Facility to build table records. It routes accounting data to multiple tables by compiling a module for each table. CA Datacom/DB stores these modules in a special table, named PRM, and loads them when it opens each accounting table.

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When to Use

When creating an accounting database or updating an existing accounting database definition, use the ACCT CATALOG option to compile accounting table definitions and initialize the Accounting Facility PRM table.

PRM Table Requirements

You must create and load the PRM table before the Accounting Facility can be run. The PRM table must be in the database that is specified in the Multi-User startup option ACCTPRM. For ease of maintenance, we recommend that the PRM table not be in the same database as the actual accounting tables. You can identify the database containing the PRM table by specifying the DBID= parameter. This has a default of DBID=4.

Validating the Database Definition

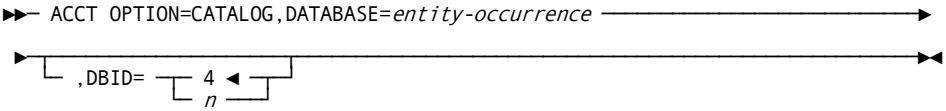
The accounting database that you are cataloging must be in PRODuction status in CA Datacom/DB. If not, an error results. The CATALOG option validates the accounting database before cataloging it in the PRM table. The EDIT option of accounting also performs a validation. Run EDIT before placing a database definition in PRODuction status. See ACCT EDIT (Edit Accounting Database Definitions) (see page 75) and the *CA Datacom/DB Database and System Administration Guide* for further details.

DSF Modules

This option uses the Datadictionary Service Facility (DSF). Therefore, be sure that the CA Datacom/DB load modules are either in the CA Datacom/DB load library or that the CA Datacom/DB load library is concatenated to the available load libraries for this execution.

How to Use

Multi-User mode is required to execute this command. To CATALOG an accounting database to the Accounting Facility, execute DBUTLTY with the command:



Command

ACCT

Invokes the Accounting Facility maintenance function.

Required Keywords

OPTION=CATALOG

Specifies the CATALOG option.

,DATABASE=

Enter the entity-occurrence name that identifies the database containing the accounting database tables to be cataloged. The CATALOG option always uses the PRODuction status version of this database in your CA Datacom/DB.

Valid Entries:

A 1- to 32-character entity-occurrence name

Default Value:

(No default)

Optional Keyword

,DBID=

Enter the database ID of the database containing the PRM table.

Valid Entries:

Valid DATACOM-ID of existing PRM database

Default Value:

4

Example JCL

The following shows the command to catalog the accounting database.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'CXX',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXXnnn,'prm.db.index',,DA                 IXX for database containing PRM
 // EXTENT SYSnnn,volser
 // DLBL PRMnnn,'prm.db',,DA                        Area containing PRM
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          ACCT  OPTION=CATALOG,DATABASE=ACCOUNTING
 /*
 /&
 * $$ EOJ
```

Sample Report: ACCT CATALOG

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          ACCT OPTION=CATALOG,DATABASE=ACCOUNTING

                    FUNCTION=ACCT
          ❷            DATABASE=ACCOUNTING
                       OPTION=CATALOG


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                        ACCOUNTING CATALOG OPTION FOR TABLE ACCOUNTING-A00(PROD) IN DATABASE   ❹
DBID OF ACCOUNTING BASE:     5 }
DATABASE TABLE NAME: A00       }
DATABASE KEY NAME: A00KY       }  ❺
DATADICTIONARY KEY NAME: A00KY }
    STARTING AT OFFSET 00000 FOR A LENGTH OF 012 WITH DD NAME OF A00KY    ❻
ELEMENT NAMES USED:
    DATE STARTING AT OFFSET 00024 FOR A LENGTH OF 008 WITH DD NAME OF DATE   }
    ETIME STARTING AT OFFSET 00036 FOR A LENGTH OF 008 WITH DD NAME OF ETIME }
    JNAME STARTING AT OFFSET 00000 FOR A LENGTH OF 008 WITH DD NAME OF JNAME }
    LOGIO STARTING AT OFFSET 00064 FOR A LENGTH OF 004 WITH DD NAME OF LOGIO }
    REQS  STARTING AT OFFSET 00060 FOR A LENGTH OF 004 WITH DD NAME OF REQS  }
    RTIME STARTING AT OFFSET 00044 FOR A LENGTH OF 008 WITH DD NAME OF RTIME }   ❼
    RUNIT STARTING AT OFFSET 00008 FOR A LENGTH OF 004 WITH DD NAME OF RUNIT }
    SDATE STARTING AT OFFSET 00012 FOR A LENGTH OF 006 WITH DD NAME OF SDATE }
    STIME STARTING AT OFFSET 00018 FOR A LENGTH OF 006 WITH DD NAME OF STIME }
    TIME  STARTING AT OFFSET 00030 FOR A LENGTH OF 006 WITH DD NAME OF TIME  }
    WTIME STARTING AT OFFSET 00052 FOR A LENGTH OF 008 WITH DD NAME OF WTIME }
STORAGE STATISTICS:
    NUMBER OF BYTES PER TABLE ENTRY: 102                                    }
    AMOUNT OF OUTPUT RECORD USED: 1926 BYTES                                }
    MINIMUM NUMBER OF TABLE ENTRIES: 019                                    }   ❽
    MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS: 11    }
DB13211I - ACCOUNTING DEFINITIONS FOR TABLE A00 NOW IN PRODUCTION STATUS  ❾
```

❹

Message displaying table name or database name and version/status being cataloged.

❺

Accounting database ID defined to CA Datacom/DB, followed by the table name and key name defined to CA Datacom/DB.

❻

Length and offset for the specified key.

❼

List of the elements in the table. One line for each element, identifying each element's offset and length.

❽

Storage Statistics:

NUMBER OF BYTES PER TABLE ENTRY

Number of bytes in the internal accounting buffers required to hold one record from this table.

AMOUNT OF OUTPUT RECORD USED

Number of bytes in the PRM table record required to contain the compiled instructions to build this table.

MINIMUM NUMBER OF TABLE ENTRIES

Assuming a size of 4096 for the buffer size specified in the Multi-User startup option ACCT, count of the minimum number of entries for this table that can be held in the internal accounting buffer.

MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS

The minimum value required in the MAXELRQ parameter based on the number of elements actually used.

❾

Message indicating that the definitions for this accounting table have been placed in PRODuction status.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                           ACCOUNTING CATALOG OPTION FOR TABLE ACCOUNTING-A02(PROD) IN DATABASE
DBID OF ACCOUNTING BASE:     5
DATABASE TABLE NAME: A02
CONDITIONAL EXPRESSION:
                              (TINDX="FF")  ❿
DATABASE KEY NAME: A02KY
DATADICTIONARY KEY NAME: A02KY
    STARTING AT OFFSET 00000 FOR A LENGTH OF 075 WITH DD NAME OF A02KY
ELEMENT NAMES USED:
    BASE  STARTING AT OFFSET 00000 FOR A LENGTH OF 002 WITH DD NAME OF BASE
    EXCPS STARTING AT OFFSET 00083 FOR A LENGTH OF 004 WITH DD NAME OF EXCPS
    REQS  STARTING AT OFFSET 00075 FOR A LENGTH OF 004 WITH DD NAME OF REQS
    SETCT STARTING AT OFFSET 00079 FOR A LENGTH OF 004 WITH DD NAME OF SETCT
    SETKY STARTING AT OFFSET 00005 FOR A LENGTH OF 070 WITH DD NAME OF SETKY
    TABLE STARTING AT OFFSET 00002 FOR A LENGTH OF 003 WITH DD NAME OF TABLE
STORAGE STATISTICS:
    NUMBER OF BYTES PER TABLE ENTRY: 121
    AMOUNT OF OUTPUT RECORD USED: 1673 BYTES
    MINIMUM NUMBER OF TABLE ENTRIES: 019
    MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS: 06
DB13211I - ACCOUNTING DEFINITIONS FOR TABLE A02 NOW IN PRODUCTION STATUS
```

❿

Display of the conditional expression that limits the accounting data that goes to this table. In this specific example, TINDX="FF" specifies that SELFR commands which cause a temporary index to be built have been selected to be recorded in this table.

ACCT CLOSE (Close Accounting Tables)

Accounting tables specified in the ACCT Multi-User startup option (see the *CA Datacom/DB Database and System Administration Guide*) are opened by the Accounting Facility when it initializes, unless their status is specified as CLOSED on the startup option. Each accounting table can be selectively closed with a utility console command.

When to Use

Use the ACCT CLOSE option to terminate recording of data in a specific accounting table. You can have some accounting tables defined that are only being used to record data during specific times of the day or for special situations. The Accounting Facility spills all active records from memory before closing the table.

You can use the CLOSE option to stop the Accounting Facility by closing all of the accounting tables. Once the last table closes, the Accounting Facility becomes dormant, closing the accounting database. You must do this to make any change to the accounting database structure.

The accounting databases are subject to the access restriction described in ACCESS (Restricting Opens or Maintenance) (see page 55). Disabling an accounting database from being opened does not, however, cause it to be closed.

Length of Time to Process

Some time can elapse between requesting a CLOSE and the actual closing of the table depending on the number of records in memory that must be spilled to the table.

Multi-User Facility Connection

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

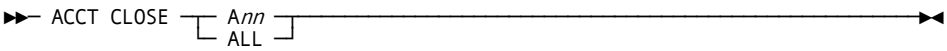
Scheduling OPENs and CLOSEs

A request to CLOSE a table that is not open is ignored by the Accounting Facility. Because the OPEN and CLOSE options are scheduled asynchronously, a CLOSE followed immediately by an OPEN can result in the table not being closed. If the table CLOSE has not started by the time the OPEN is received, the table is not closed (see ACCT OPEN (Open Accounting Tables) (see page 81)).

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

ACCT

Invokes the Accounting Facility maintenance function.

CLOSE

Desired action

Required Keywords

A*nn* or ALL

Specifies the accounting table name. Entering **ALL** indicates a global close of all accounting tables and stops the Accounting Facility.

Valid Entries:

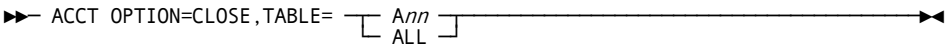
A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. To CLOSE an accounting database table in the Accounting Facility, execute DBUTLTY using the command:



Command

ACCT

Invokes the Accounting Facility maintenance function.

Required Keywords

OPTION=CLOSE

Specifies the CLOSE option.

,TABLE=

Specifies which table in the accounting database is to be spilled and closed. Entering **ALL**, indicates all tables are to be closed and the Accounting Facility is to be stopped.

Valid Entries:

A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

Example JCL

The following shows the command to close all accounting tables.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          ACCT     OPTION=CLOSE,TABLE=ALL
 /*
 /&
 * $$ EOJ
```

Sample Report: ACCT CLOSE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          ACCT OPTION=CLOSE,TABLE=ALL

                    FUNCTION=ACCT
          ❷            OPTION=CLOSE
                       TABLE=ALL


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

DB13105I - ACCT                CLOSE    REQUEST ACCEPTED  ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User Facility communication queue.

ACCT EDIT (Edit Accounting Database Definitions)

The ACCT EDIT function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When to Use

Before an accounting database can be cataloged, many special edits of the accounting table definitions must be performed. These edits are performed by the ACCT EDIT option. They are also automatically performed by the ACCT CATALOG option. EDIT can be requested for TEST or PRODuction status versions of the accounting database structure in CA Datacom/DB.

Database Definitions in TEST Status

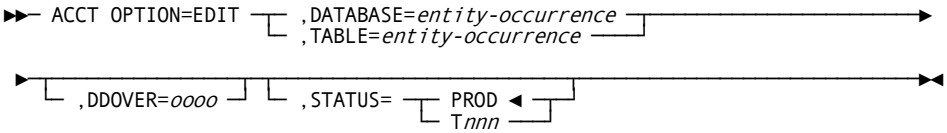
The EDIT option is recommended when building an accounting database for the first time. Since EDIT can be run against a TEST status database definition in CA Datacom/DB, you can validate the structure, locating any errors before placing the definition in PRODuction status. EDIT is also recommended when changing the accounting database structure.

DSF Modules

This option uses the Datadictionary Service Facility (DSF). Therefore, be sure that the CA Datacom/DB load modules are either in the CA Datacom/DB load library or that the CA Datacom/DB load library is concatenated to the available load libraries for this execution.

How to Use

Multi-User mode is required to execute this command. To EDIT an accounting database definition, execute DBUTLTY using the command:



Command

ACCT

Invokes the Accounting Facility maintenance function.

Required Keywords

OPTION=EDIT

Specifies the EDIT option.

Note: You must specify one of the following two keywords. If the entire database is to be edited, specify the DATABASE keyword. If only one table is to be edited, specify the TABLE keyword.

,DATABASE=

Specifies the entity-occurrence name for the database containing the accounting database definition to be edited. The keywords DATABASE and TABLE are mutually exclusive. Specify either DATABASE or TABLE, but not both.

Valid Entries:

1- to 32-character entity-occurrence name for the accounting database

Default Value:

(No default)

,TABLE=

Specifies the name of an individual accounting table to be edited. The keywords DATABASE= and TABLE= are mutually exclusive, that is to say, specify either DATABASE= or TABLE=, but not both.

Valid Entries:

1- to 32-character accounting table name

Default Value:

(No default)

Optional Keywords

,DDOVER=

Specifies the CA Datacom/DB override code for the database, if it is secured with CA Datacom/DB security.

Valid Entries:

A 4-charactdader CA Datacom/DB security override code

Default Value:

(No default)

,STATUS=

Specifies the status of the database definition to be edited.

- If the database is in TEST status, enter the 4-character alphanumeric CA Datacom/DB version number in the form Tnnn.
- If the database is in PRODuction status in your CA Datacom/DB, enter PROD.

Valid Entries:

Tnnn or PROD

Default Value:

PROD

Example JCL

The following shows the command to perform the EDIT option on the accounting database.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'CXX',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXXnnn,'prm.db.index',,DA                 IXX for database containing PRM
 // EXTENT SYSnnn,volser
 // DLBL PRMnnn,'prm.db',,DA                        Area containing PRM
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
         ACCT  OPTION=EDIT,DATABASE=ACCOUNTING
 /*
 /&
 * $$ EOJ
```

Sample Report: ACCT EDIT

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          ACCT OPTION=EDIT,DATABASE=ACCOUNTING

                    FUNCTION=ACCT
          ❷            DATABASE=ACCOUNTING
                       OPTION=EDIT


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

                        ACCOUNTING EDIT OPTION FOR TABLE ACCOUNTING-A00(PROD) IN DATABASE  ❹
DBID OF ACCOUNTING BASE:     5 }
DATABASE TABLE NAME: A00       }
DATABASE KEY NAME: A00KY       }  ❺
DATADICTIONARY KEY NAME: A00KY }
    STARTING AT OFFSET 00000 FOR A LENGTH OF 012 WITH DD NAME OF A00KY    ❻
ELEMENT NAMES USED:
    DATE  STARTING AT OFFSET 00024 FOR A LENGTH OF 006 WITH DD NAME OF DATE  }
    ETIME STARTING AT OFFSET 00036 FOR A LENGTH OF 008 WITH DD NAME OF ETIME }
    JNAME STARTING AT OFFSET 00000 FOR A LENGTH OF 008 WITH DD NAME OF JNAME }
    LOGIO STARTING AT OFFSET 00064 FOR A LENGTH OF 004 WITH DD NAME OF LOGIO }
    REQS  STARTING AT OFFSET 00060 FOR A LENGTH OF 004 WITH DD NAME OF REQS  }
    RTIME STARTING AT OFFSET 00044 FOR A LENGTH OF 008 WITH DD NAME OF RTIME }   ❼
    RUNIT STARTING AT OFFSET 00008 FOR A LENGTH OF 004 WITH DD NAME OF RUNIT }
    SDATE STARTING AT OFFSET 00012 FOR A LENGTH OF 006 WITH DD NAME OF SDATE }
    STIME STARTING AT OFFSET 00018 FOR A LENGTH OF 006 WITH DD NAME OF STIME }
    TIME  STARTING AT OFFSET 00030 FOR A LENGTH OF 006 WITH DD NAME OF TIME  }
    WTIME STARTING AT OFFSET 00052 FOR A LENGTH OF 008 WITH DD NAME OF WTIME }
STORAGE STATISTICS:
    NUMBER OF BYTES PER TABLE ENTRY: 102                                    }
    AMOUNT OF OUTPUT RECORD USED: 1926 BYTES                                }
    MINIMUM NUMBER OF TABLE ENTRIES: 019                                    }   ❽
    MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS: 11    }
DB13211I - ACCOUNTING DEFINITIONS FOR TABLE A00 NOW IN PRODUCTION STATUS  ❾
```

❹

Message displaying table name or database name and version/status being cataloged.

❺

Accounting database ID defined to CA Datacom/DB, followed by the table name and key name defined to CA Datacom/DB.

❻

Length and offset for the specified key.

❼

List of the elements in the table. One line for each element, identifying each element's offset and length.

❽

Storage Statistics:

NUMBER OF BYTES PER TABLE ENTRY

Number of bytes in the internal accounting buffers required to hold one record from this table.

AMOUNT OF OUTPUT RECORD USED

Number of bytes in the PRM table record required to contain the compiled instructions to build this table.

MINIMUM NUMBER OF TABLE ENTRIES

Assuming a buffer size of 4096 specified in the Multi-User startup option ACCT, count of the minimum number of entries for this table that can be held in the internal accounting buffer.

MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS

The minimum value required in the Multi-User startup option MAXELRQ based on the number of elements actually used.

❾

Message indicating that the definitions for this accounting table have been placed in PRODuction status.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                            ACCOUNTING EDIT OPTION FOR TABLE ACCOUNTING-A02(PROD) IN DATABASE
DBID OF ACCOUNTING BASE:     5
DATABASE TABLE NAME: A02
CONDITIONAL EXPRESSION:
                              (TINDX="FF")       ❿
DATABASE KEY NAME: A02KY
DATADICTIONARY KEY NAME: A02KY
    STARTING AT OFFSET 00000 FOR A LENGTH OF 075 WITH DD NAME OF A02KY
ELEMENT NAMES USED:
    BASE  STARTING AT OFFSET 00000 FOR A LENGTH OF 002 WITH DD NAME OF BASE
    EXCPS STARTING AT OFFSET 00083 FOR A LENGTH OF 004 WITH DD NAME OF EXCPS
    REQS  STARTING AT OFFSET 00075 FOR A LENGTH OF 004 WITH DD NAME OF REQS
    SETCT STARTING AT OFFSET 00079 FOR A LENGTH OF 004 WITH DD NAME OF SETCT
    SETKY STARTING AT OFFSET 00005 FOR A LENGTH OF 070 WITH DD NAME OF SETKY
    TABLE STARTING AT OFFSET 00002 FOR A LENGTH OF 003 WITH DD NAME OF TABLE
STORAGE STATISTICS:
    NUMBER OF BYTES PER TABLE ENTRY: 121
    AMOUNT OF OUTPUT RECORD USED: 1673 BYTES
    MINIMUM NUMBER OF TABLE ENTRIES: 019
    MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS: 06
DB13212I - NO EDITING ERRORS FOUND FOR ACCOUNTING TABLE A02
```

❿

Display of the conditional expression that limits the accounting data that goes to this table. In this specific example, TINDX="FF" specifies that SELFR commands which cause a temporary index to be built have been selected to be recorded in this table.

ACCT OPEN (Open Accounting Tables)

Accounting tables specified in the ACCT Multi-User startup option, described in the *CA Datacom/DB Database and System Administration Guide,* are opened by the Accounting Facility when it initializes, unless their status is specified as CLOSED on the startup option. The ACCT OPEN function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

When to Use

Execute the ACCT OPEN option to open accounting tables that either were not opened when the Accounting Facility initialized or were closed with the ACCT CLOSE option.

If an accounting table is only used to record data during specific times of the day or for special situations, use the ACCT OPEN option to begin recording data in it.

After closing an accounting table to spill data and delete records for week-end or month-end processing or to perform maintenance to the table definition, use the OPEN option to reactivate a table.

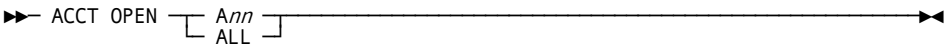
Scheduling OPENs and CLOSEs

A request to OPEN a table that is already open is ignored by the Accounting Facility. Because the OPEN and CLOSE options are scheduled asynchronously, an OPEN followed immediately by a CLOSE can result in the table not being opened. If the table OPEN has not started by the time the CLOSE is received, the table is not opened (see ACCT CLOSE (Close Accounting Tables) (see page 71)).

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

ACCT

Invokes the Accounting Facility maintenance function.

OPEN

Requested action.

Required Keywords

A*nn* or ALL

Specifies the table you want to open. Entering **ALL** specifies a global open of all accounting tables.

Valid Entries:

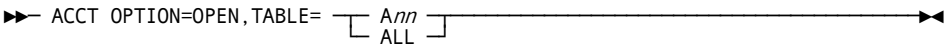
A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. To OPEN an accounting database table in the Accounting Facility, execute DBUTLTY using the following command format.



Command

ACCT

Invokes the Accounting Facility maintenance function.

Required Keywords

OPTION=OPEN

Specifies the OPEN option.

,TABLE=

Specifies the table in the accounting database that is to be opened. Entering **ALL**, indicates all accounting tables are to be opened.

Valid Entries:

Ann (where nn is two numeric digits) or ALL

Default Value:

(No default)

Example JCL

The following shows the command to OPEN table A01 in the accounting database.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
            ACCT     OPTION=OPEN,TABLE=A01
 /*
 /&
 * $$ EOJ
```

Sample Report: ACCT OPEN

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          ACCT OPTION=OPEN,TABLE=A01

                    FUNCTION=ACCT
          ❷            OPTION=OPEN
                       TABLE=A01


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - ACCT                 OPEN     REQUEST ACCEPTED    ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User Facility communication queue.

ACCT SPILL (Force Accounting Data Recording)

The ACCT SPILL function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

When to Use

The Accounting Facility temporarily accumulates data in memory for all accounting tables in order to minimize input/output event overhead. Because data is held in memory, the accounting data in the database does not always reflect the latest information. Use the SPILL option to force the latest information to be recorded before running any reports or queries against the accounting database.

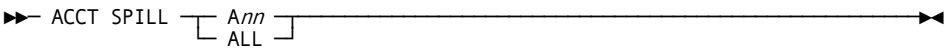
Length of Time to Process

Some time can elapse between requesting a SPILL and receiving acknowledgement of completion of the request, depending on the number of records in memory that must be spilled to the table.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

ACCT

Invokes the Accounting Facility maintenance function.

SPILL

Desired action on the designated table(s).

Required Keywords

A*nn* or ALL

Specifies the particular table to spill. Entering **ALL** indicates a global spill of all accounting tables.

Valid Entries:

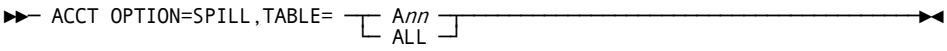
A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. To SPILL an accounting database table, execute DBUTLTY using the command:



Command

ACCT

Invokes the Accounting Facility maintenance function.

Required Keywords

OPTION=SPILL

Specifies the SPILL option.

,TABLE=

Specifies the table in the accounting database that is to be spilled. Entering **ALL** indicates all tables are to be spilled.

Valid Entries:

A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

Example JCL

The following shows the command to SPILL table A09 in the accounting database.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
           ACCT     OPTION=SPILL,TABLE=A09
 /*
 /&
 * $$ EOJ
```

Sample Report: ACCT SPILL

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                     CONTROL CARD(S)
                     .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          ACCT OPTION=SPILL,TABLE=A09

                     FUNCTION=ACCT
          ❷              OPTION=SPILL
                        TABLE=A09


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - ACCT                SPILL    REQUEST ACCEPTED    ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User Facility communication queue.

Chapter 5: BACKUP (Create Backups)

Making backups of critical data is essential in a data processing installation. DBUTLTY provides functions to create backups of either the Directory (CXX), any database, or any data area within a database.

Make frequent backups of the Directory (CXX). If a hardware malfunction occurs during maintenance with the CXXMAINT function, you can re-create the Directory using the backup. You can make a backup of the Directory (CXX) at any time.

A backup of the Directory is the first step in moving the Directory to a different disk drive or to a disk drive of a different device type. See BACKUP CXX (Back Up Directory) (see page 89) and LOAD CXX (Build CXX) (see page 386) for details.

DBUTLTY provides several options for creating a backup of a data area. Creating a backup in Native Key sequence is the first step in reorganizing a data area. See the *CA Datacom/DB Database and System Administration Guide* for details on reorganizing a data area or loading a multiple table area.

For information about DBUTLTY STATUS console command support for this function, see COMM STATUS (Print Status of Multi-User Jobs).

More information about the impact of table partitioning on the BACKUP function is in The BACKUP Utility (see page 40).

BACKUP CXX (Back Up Directory)

The BACKUP CXX option backs up the Directory (CXX).

When to Use

Use the BACKUP function to generate a backup of the Directory (CXX):

- To use for rebuilding the Directory in case of hardware errors or accidental destruction.
- Before running the CXXMAINT CONVERT option for upgrading the CXX to the present version of CA Datacom/DB, read CXXMAINT CONVERT1112 (Upgrade CXX to Current Version) (see page 238).
- As the first step in moving the Directory to either a larger extent or different disk device type.

Backup Output

The BACKUP function uses a variable record format and a block size equal to 4K. The backup output file can be on a disk. However, the backup stored on the disk is in a special format. If a hardware malfunction occurs and the Directory is not accessible, you cannot point to the backup on disk and continue to function. You must LOAD the Directory from the backup to obtain a valid Directory.

Databases Accessed by SQL Statements

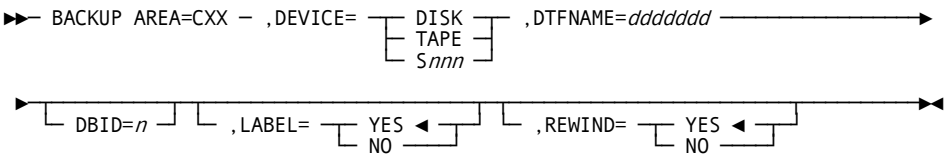
If your site uses SQL, backing up the Directory without backing up the DATA-DICT and Data Definition Directory (DDD) databases can result in loss of synchronization which is critical for SQL operations. To retain synchronization when you back up the Directory, you must back up the DATA-DICT database and the DDD database.

Note: See the *CA Datacom/DB SQL User Guide* for information about the SQL Object Consistency Analyzer. You can use the SQL Object Consistency Analyzer to determine whether CA Datacom/DB, Data Definition Directory, Schema Information Tables areas, and the Directory are synchronized and to determine the steps needed to correct any problems.

How to Use

You can execute this command in either Single User or Multi-User mode. To back up your Directory, execute DBUTLTY using the command:

Back Up Directory



Command

BACKUP

Invokes the BACKUP function.

Required Keywords

AREA=CXX

Identifies the area to be backed up as the Directory.

DEVICE=

Specifies the device type for the output data set.

When DEVICE=TAPE is specified, the system logical unit (SYS) number must be 5 (SYS005).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the output file. If DEVICE=TAPE, this name must match the corresponding name which you code in your z/VSE JCL TLBL statement; otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

Optional Keywords

DBID=

Identifies the database to be backed up. If not specified, the function backs up the entire CXX, as would have occurred in the previous release, before this optional DBID= keyword was added. With DBID= specified, the only output is the designated database. Only one DBID can be specified and only once, that is, you cannot specify a range of databases.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

LABEL=

If DEVICE=TAPE (either TAPE or the Snnn form), LABEL= specifies whether tape labels are to be processed. We recommend that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

REWIND=

If DEVICE=TAPE (either TAPE or the Snnn form), YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

Example JCL (Output to Tape)

The following shows the command to back up the Directory (CXX) to the output data set OUT3 on tape.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL OUT3,'cxx.bkup'                Output data
 // EXEC DBUTLTY,SIZE=750K
         BACKUP   AREA=CXX,DEVICE=TAPE,DTFNAME=OUT3
 /*
 /&
 * $$ EOJ
```

Example JCL (Output to Disk)

The following shows the command to back up the Directory (CXX) to the output data set OUT3 on disk.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT ,volser
 // DLBL OUT3,'cxxbkup',n,SD                        output data
 // EXTENT ,volser,1,0,strk,ntrks
 // EXEC DBUTLTY,SIZE=750K
           BACKUP   AREA=CXX,DEVICE=DISK,DTFNAME=OUT3
 /*
 /&
 * $$ EOJ
```

Sample Report: BACKUP CXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          BACKUP AREA=CXX,DTFNAME=OUT3

                    FUNCTION=BACKUP
          ❷            AREA=CXX
                       DTFNAME=OUT3
          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                   Copyright © 2009 CA. All rights reserved.                  *               SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹         *                                    REQUEST COMPLETE
```

❹

Confirmation that the request completed correctly. Any errors encountered during processing are noted with messages here.

Back Up Data Area

The BACKUP Data Area option of DBUTLTY reads the specified area (or all areas of a database for a "base-level backup") and writes a sequential output data set. A backup can also be made of a data area that either INCLUDEs or EXCLUDEs specific tables. The output data set record format is variable blocked.

Note: Since a MRDF-supported virtual area exists only in the Multi-User Facility memory, you cannot perform a backup of the data without MULTUSE=YES. If you execute the backup of a virtual table without MULTUSE=YES, the backup accesses the disk area either unloading no records or unloading any records as they existed in the area before it became virtual. You can perform a backup of a covered area since CA Datacom/DB writes to both the area in memory and the DASD copy.

When to Use

Use the BACKUP data area option to:

- Create an unloaded copy of a data area for input to the LOAD function to reorganize the data area in Native Key sequence.
- Create a backup of a table that is to be moved to another data area after appropriate changes to CA Datacom/DB definitions.
- Create a backup of the data area that can be used in a future forward recovery operation if the data area becomes damaged.
- Retrieve the data in a data area after the Index has become damaged.

Native Sequence Backups

The native sequence backup uses the Native Key defined to each table in a data area to reorganize the data area in Native Key sequence. You cannot use the native sequence for a backup if the Native Key is also defined as the direct access key.

A native sequence backup of a data area is not allowed if the database is open for update by any other task. Once a Native Sequence backup begins, CA Datacom/DB does not allow an open for update.

With a Native Key sequence, the FIRSTKEY= and LASTKEY= optional keywords can be used to selectively include or exclude data from BACKUP output file. This can be used for building test data, purging old data, or selecting segments of CA Datacom tables for offline processing.

Physical Sequence Backups (Not Continuous Operation)

Create a physical sequence backup for later use in restoring a damaged database. A noncontinuous operation backup (UPDATE=YES) in physical sequence can be made only if the database is not open for update.

Backups During Continuous Operation

To perform a backup while processing continues, specify a physical sequence backup and UPDATE=NO in the command. The UPDATE=NO parameter allows the database to be updated or an update job may start during the backup. You are not blocked from performing an INIT or LOAD function during the backup. Allowing maintenance during a backup can result in backup data that cannot be used without a forward recovery. (See RECOVERY (Rebuild a Database) (see page 443) and the *CA Datacom/DB Database and System Administration Guide* for details about recovery.)

Compression

If CA Datacom/DB compression or user compression is being used, the data can be written expanded or compressed. If the backup is being done with a following load which changes the type of compression, the backup should be expanded. Backing up compressed data generally runs faster.

Effects of Reorganization on Recovery

The forward and backward recovery operations require that record location information in the database exactly match the record location information that existed at the time the maintenance was performed. A reorganization of the database causes new record location information to be assigned. Therefore, any reorganization causes Recovery File(s) (RXX) created prior to the reorganization to be unusable.

Note: Reorganization occurs during a backup when RECID=YES is not specified.

Backup for Forward Recovery

To use forward recovery, you must restore the records in the database to the same locations at which they resided at the time the recovery records were logged. A backup containing the records that can be restored correctly can be obtained in one of two ways:

- Create a backup in physical sequence that contains record location information (RECID=YES).
- Create a backup in Native Key sequence or physical sequence with RECID=NO and immediately restore it. This reorganization invalidates any Recovery Files (RXX) that were made prior to the backup. However, you can use any Recovery Files created beyond this point to restore the database after reloading this backup.

Non-DBUTLTY Backups

A physical sequence backup can also be obtained using any other physical backup utility. The DBUTLTY LOAD function does not accept a non-DBUTLTY backup as input. If the Multi-User Facility was active and the database was not disabled when restoring data from this type of backup, the Index must be rebuilt using the DBUTLTY RETIX (Index Retrieve) function.

Backing Up Compressed Data Areas (Physical Sequence, UPDATE=NO)

When backing up compressed data areas, if the area is being updated, records may move from one disk location to another and can be duplicated or lost. To prevent this from occurring, use the DBUTLTY LOCK and UNLOCK functions. The LOCK function prevents records from moving until the UNLOCK is issued. First, issue the lock, run the backup, and then issue the unlock. UPDAT requests that cause records to move wait for the unlock. See LOCK (Lock a Data Area) (see page 423) for more information.

This procedure assumes one Multi-User Facility that is enabled when the lock is issued. If there is more than one Multi-User Facility that can open the area for update, lock each Multi-User Facility. *No* protection is provided if a Multi-User Facility is disabled at the time of the lock, then subsequently enables and opens the area for update while the backup is in process.

Restoring a Damaged Index

If the Index becomes damaged or cannot be used, then a backup in physical sequence provides a means of re-creating the area without accessing the Index. You can use a physical sequence backup of a data area even if the Directory (CXX) entries describing the database were destroyed and rebuilt.

Backup Format

The backup format is not the same as that of the CA Datacom/DB area itself, and you cannot use it except as input to a data load.

I/O or other errors result if you attempt to use a physical sequence backup with record location information (RECID=YES) as the source for a load of data:

- To a different device type.
- To a smaller extent of the same device type.
- To a different block size.
- When concatenated input files are used.
- If the REMAP= keyword of the LOAD function is used.
- If table partitioning is being used, and data records are moved to new partitions.
- If loading compressed data as uncompressed or uncompressed as compressed.

Master List Requirements

The data area BACKUP function uses a form of GSETL/GETIT or GSETP/GETPS command for reading the data blocks. This includes reading multiple blocks per I/O. You can allocate sequential buffers with the SEQBUFS= keyword.

Table IDs

The Table IDs are written during backup as part of the output sequential records. CA Datacom/DB uses these Table IDs during the LOAD function to identify the record. The Table ID cannot differ from the definition in force at the time of the LOAD using the default LOADID=YES. If the table IDs differ, LOAD may match on table names using LOADID=NO.

Databases Accessed by SQL Statements

If your site uses SQL, backing up the data areas of either the DATA-DICT or Data Definition Directory (DDD) databases without backing up the Directory can result in loss of synchronization which is critical for SQL operations. To retain synchronization, you must back up the DATA-DICT database, the DDD database, and the Directory.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO and either SEQ=NAT or SEQ=PHY,UPDATE=YES*

- The database may *not* be open for update anywhere.
- The database may or may not be open in any Multi-User Facility for read.
- The database may or may not have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO and either SEQ=NAT or SEQ=PHY,UPDATE=YES*

- The database will be opened for *unloading* status.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when SEQ=PHY (UPDATE=NO default)*

- The database may or may not be open update or read or *unloading* anywhere.
- The database will not be opened for read or update or *unloading.*
- This option has no requirements, it provides no protection.

Environmental **Requirements** *when MULTUSE=YES*

- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/READ/UTLTY.
- The area ACCESS must be set WRITE/READ/UTLTY.
- The area may have no table in an open URT with UPDATE=YES.
- The area may *not* have any other MULTUSE=YES utility executing.
- When MULTUSE=YES, this function is not allowed against a Data Sharing Multi-User Facility if more than one is executing.

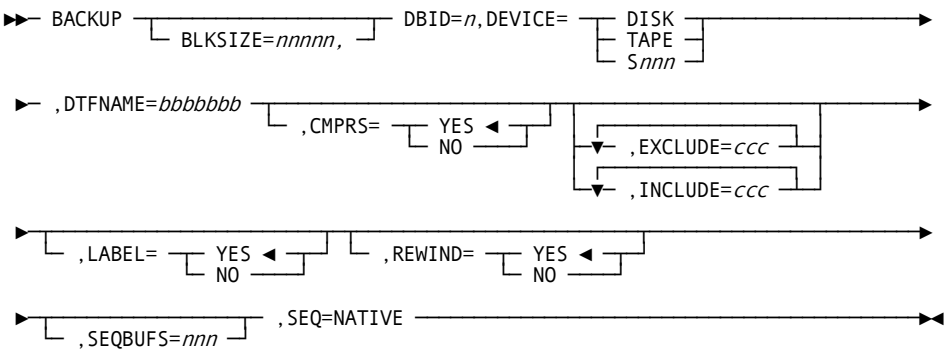
Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The database will have no *unloading* status set.
- The utility sets ACCESS area READ (utility set).
- The utility sets BACKUP executing this Multi-User Facility this database area.

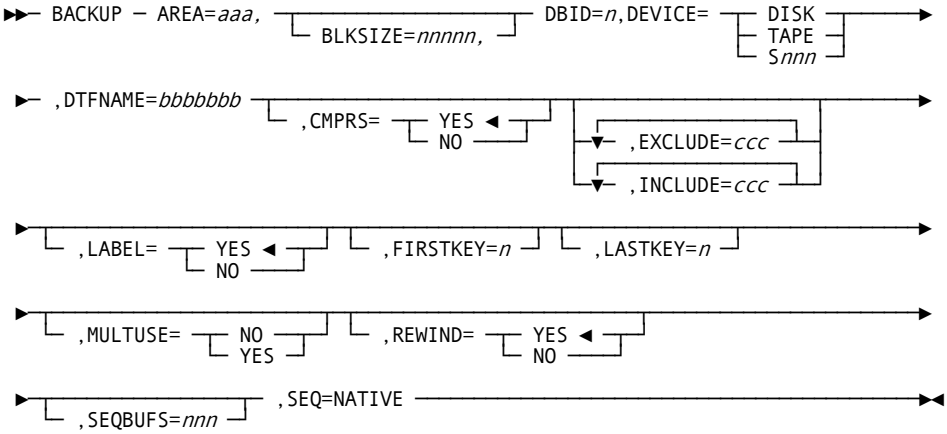
How to Use

You can execute this command in either Single User or Multi-User mode. To create a backup of a data area, execute DBUTLTY using the following command. You can code multiple EXCLUDE= or INCLUDE= keywords.

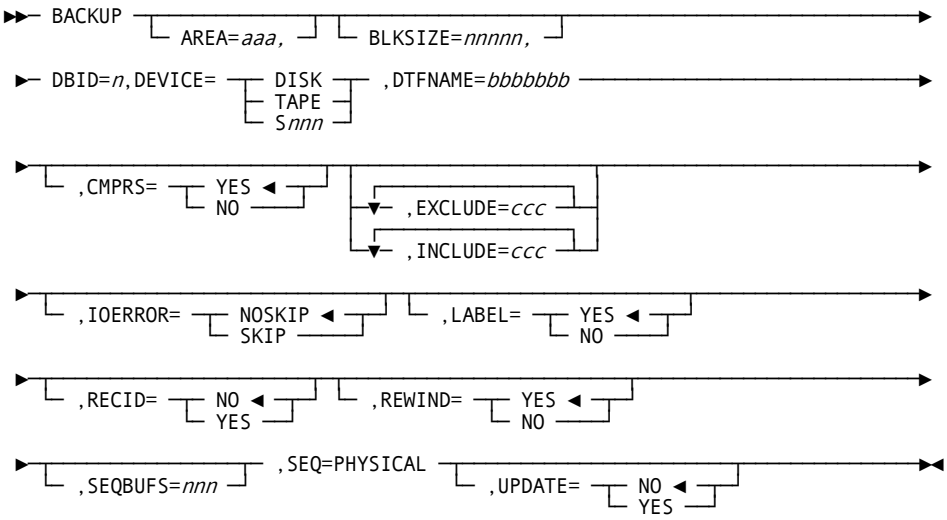
Back Up All Areas in Database (z/VSE)



Back Up Single Area in Database (z/VSE)



Back Up Data Area Physical (z/VSE)



Command

BACKUP

Invokes the BACKUP function.

Required Keywords

DBID=

Identifies the database for the area to be read.

Valid Entries:

Valid DATACOM-ID of an existing database

Default Value:

(No default)

DEVICE=

Specifies the device type for the output data set.

When DEVICE=TAPE is specified, the system logical unit (SYS) number must be 5 (SYS005).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

DTFNAME=

Specifies the JCL TLBL/DLBL name for the output data set. If DEVICE=TAPE, this name must match the name which you code in your JCL TLBL statement; otherwise, this name must match the name you code in your JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

Optional Keywords

AREA=

Identifies the data area to be backed up. If you omit this keyword, each area in the database is opened, backed up, and closed, in turn.

Note: If you specify MULTUSE=, you must also specify an area name.

Valid Entries:

A 3-character name of an area

Default Value:

(No default)

BLKSIZE=

Indicates the block size for the output data set. For best performance, use 32767 when going to tape and the largest block size which uses all of a track when going to disk.

Note: If the block size is not large enough to contain the largest record in the area (see the following formula), DBUTLTY terminates abnormally.

Calculate the minimum block size for the output data set as follows:

A + C + 30
where:
A is the length of the largest record in the area,
C is 0, if the data is not compressed, or
C is (A/128) + 1, if the data is compressed.

Valid Entries:

A 1- to 5-digit number

Default Value:

4096 (for DEVICE=DISK), or
32767 (for DEVICE=TAPE)

If you specify a block size with this parameter, the utility ignores any BLKSIZE parameter in the JCL. If you do not specify a block size with this parameter, the utility uses the BLKSIZE specified in the JCL if one exists. In the absence of either this parameter or a specification in the JCL, the utility uses the default value.

,CMPRS=

Specifies, for compressed tables, whether the output is to remain compressed or is to be expanded. This parameter has no effect on tables that are not defined with compression.

YES

Specifies that DBUTLTY write the data "as is" from the area to the sequential output file. That is, a compressed table remains compressed on the output data set.

NO

Specifies that DBUTLTY write the data in expanded form to the sequential output file. The NO option should be used only when the data being produced is intended to be used as input to a user-written program or if the table is to be redefined with a different combination of CA Datacom/DB compression and user compression options than the current definition.

Note: You must not specify CMPRS=NO and RECID=YES.

Valid Entries:

NO or YES

Default Value:

YES

,EXCLUDE=

Specifies the name of the table that is not to be backed up. To indicate more than one table, code the EXCLUDE= parameter more than once.

Tables that are excluded lose their data when DBUTLTY reloads from that backup. Use the EXTRACT function instead of the BACKUP function for single table recovery.

This parameter is mutually exclusive with the INCLUDE= parameter.

If neither EXCLUDE= nor INCLUDE= is specified, all the tables are backed up.

Valid Entries:

Specify a table name within the area (or database if omitting AREA=)

Default Value:

(No default)

,FIRSTKEY= and/or ,LASTKEY=

These keywords allow you to select a segment of a table or area by Native Key (SEQ=NATIVE) value to be processed for BACKUP. The BACKUP function normally defaults to include the full Native Key range from low values to high values. FIRSTKEY= overrides the default starting position while LASTKEY= overrides the default ending position, limiting the records retrieved for output to the BACKUP file. If one keyword is specified and the other is not specified, a default value is selected for the missing keyword. Normal CA Datacom key lengths are from 1 through 180 bytes. The key value data entered for FIRSTKEY= and LASTKEY= can be from 0 (zero) through 59 bytes long. You only need to code the number of positions of the key value that are significant to you, but be aware that the value coded is *left justified regardless of key field data type.* The utility pads the low order or remaining positions of the key value with low values on FIRSTKEY= and high values on LASTKEY=. To back up the entire area in one step, specify neither FIRSTKEY= nor LASTKEY=. Alternately, you could specify FIRSTKEY=00 and/or LASTKEY=00 which causes processing to start at low values and end at high values. The length of zero indicates that the padded key values (low values and high values) are to be used.

The data is not edited or interpreted. Therefore, if the key value data you need to enter is in binary, you must set up the JCL using a hexadecimal mode display and enter the key value data in hexadecimal. This data can contain blanks, commas, and any other special characters. CA Datacom/DB cannot edit this data for syntax, so the control statement can *look* invalid, especially when it includes blanks or commas or key value data containing English words. For example, FIRSTKEY=12121212121212 means the length is 12, key value is 121212121212. Another example, FIRSTKEY=10FIRSTKEY10 means the length is 10, key value is FIRSTKEY10.

Valid Entries:

A 2-byte length (in the range of 00 through 59) followed by a character string of key value bytes (for the length specified) that represents the key value.

Default Value:

The starting/ending position of the full Native Key range from low values to high values.

,INCLUDE=

Specifies the name of a table to be backed up. To indicate more than one table, code the INCLUDE= parameter more than once.

This parameter is mutually exclusive with the EXCLUDE= parameter.

If neither INCLUDE= nor EXCLUDE= is specified, the entire area is backed up.

Valid Entries:

Any table name within the area

Default Value:

(No default)

,IOERROR=

(Not valid if AREA= keyword is omitted.) Specifies how CA Datacom/DB is to handle I/O errors during a backup. Specify this option only for a physical sequence backup.

A value of SKIP causes CA Datacom/DB to issue a console message, produce a Master List dump, and skip the block in which an I/O error occurs when the utility encounters an I/O error while reading a data block.

The utility does not skip I/O errors to other data sets or the control block of the data area. When an I/O error does occur and the SKIP value is in force, the utility exits with a good completion code since this value overrides the error condition.

If an I/O error does not occur and the value SKIP is in force, DBUTLTY issues a warning message at the end of the execution. The utility cancels the job. This prevents accidental skipping of records.

In either case, the backup is a correct, complete backup. If a skip does occur during the backup, the Master List dump and warning message provide a basis for reconstructing the missing information.

Note: IOERROR=SKIP is not recommended and exists only to assist you in salvaging as much data as possible, if all backup tapes are defective. Frequent backups eliminate the risk associated with lost data and are preferable to the IOERROR option.

Valid Entries:

SKIP or NOSKIP

Default Value:

NOSKIP

,LASTKEY=

See the description of FIRSTKEY=/LASTKEY= given previously.

LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), LABEL= specifies whether tape labels are to be processed. We recommended that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,MULTUSE=

(For area level DBUTLTY control only.) If you specify MULTUSE=, you must also specify an area name, but do not specify SEQ=PHYSICAL.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

NO or YES

Default Value:

NO

,RECID=

Specifies whether the record location information is to be backed up as part of each output record.

To create a physical sequence backup that can later be used with a forward recovery operation, specify RECID=YES.

Note the following:

- You must not specify CMPRS=NO and RECID=YES.
- If you specify RECID=YES, make certain that the value for the table's RECOVERY attribute is YES in the CA Datacom/DB definition for the table.
- Not valid if SEQ=PHYSICAL is not specified.
- When converting a data area to the Unique Row Identifier (URI) format, the backup used as input to the DBUTLTY LOAD function must be created with RECID=NO.

Valid Entries:

NO or YES

Default Value:

NO

REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,SEQ=

Indicates the sequence in which the input data is to be processed and written to the backup data set. For details on retrieval sequence, see Backup Format (see page 98).

NATIVE

Specifies that DBUTLTY read the Index and retrieve the data in Native Key sequence. When using SEQ=NATIVE, you can use the FIRSTKEY=/LASTKEY= optional keywords to select only a portion of the table or area.

PHYSICAL

Specifies that DBUTLTY not read the Index. The output has no logical sequence but does follow the physical track sequence.

Note: Do not specify SEQ=PHYSICAL if you specify MULTUSE=.

Valid Entries:

NATIVE or PHYSICAL

Default Value:

NATIVE

,SEQBUFS=

If specified, SEQBUFS= must range from 0-256, even numbers only. We recommended for best performance to specify SEQBUFS=128 or SEQBUFS=256. If memory is limited and slower performance is acceptable, lower numbers should be used.

If MULTUSE= is omitted or specified as NO, the default is SEQBUFS=8. Any value specified for SEQBUFS= less than 8 is forced up to 8. Any value greater than 128 is forced to 128. If the number of data buffers (specified with the DATANO= parameter in the DBMSTLST macro) used by this DBUTLTY was specified as 192 or more, those data buffers are used and no additional memory is allocated. The SEQBUFS= value is edited and ignored (for best performance) without a larger DATANO= parameter value specification. The DATANO= buffers are also used if they are equal to or greater than the SEQBUFS= value multipled by 1.5. Otherwise, the number of SEQBUFS= is allocated as additional sequential buffers with additional memory required.

If MULTUSE=YES is specified, the default is SEQBUFS=128. If SEQBUFS= is specified as 0-8, it is rounded up to 8. If SEQBUFS= is specified as 130-134, it is rounded up to 136, in which case no read-ahead nor multi-block reads occur. If SEQBUFS= is specified as 2-128 (even numbers only), that many private sequential buffers are allocated (memory must be available) in the MUF address space. Read ahead is done. Multiple blocks are read with each I/O when possible to reduce elapsed times and CPU times with the number per I/O as one-half of the SEQBUFS. If set as 130-256, no private sequential buffers are allocated, normal existing data buffers are used in the MUF address space. This function shares those buffers with all other tasks using MUF. If set as 132-256, read ahead is done, and multiple blocks are read with each I/O to reduce elapsed times and CPU times with the number per I/O as the SEQBUFS number minus 128 and one-half of that result. The maximum blocks per read I/O is always 64.

Valid Entries:

0-256 (even numbers only)

Default Value:

128

,UPDATE=

Indicates whether the database can be open for update at the start or during the backup.

Note: Only valid if SEQ=PHYSICAL.

NO

Allows tables in the database to be updated and allows the INIT or LOAD functions on the data area during the backup.

YES

Specifies that CA Datacom/DB determines that the data area is loaded with a loaded Index and that the database is not open for update. During the backup, the database may not be opened for update.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL (Output to Tape) - Native Key Sequence

The following shows the command to back up the data area DEM in database 001 to the output data set, OUT1, and to output the data in Native Key sequence and compressed form using default values.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA               Index Area for DBID 1
 // EXTENT SYSnnn,volser
 // DLBL DEM001,'db001.dem',,DA                 Area DEM in DBID 1
 // EXTENT SYSnnn,volser
 // ASSGN SYS010,TAPE
 // PAUSE MOUNT TAPE
 // TLBL OUT1,'dem.bkup'                        Output data
 // EXEC DBUTLTY,SIZE=750K
          BACKUP    AREA=DEM,DBID=1,DEVICE=TAPE,DTFNAME=OUT1
 /*
 /&
 * $$ EOJ
```

Example JCL (Output to Disk) - Native Key Sequence

The following shows the command to backup the data area DEM in database 001 to the output data set, OUT1 on disk. Output the data in Native Key sequence and in compressed form.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
```

```
// EXTENT SYSnnn,volser
// DLBL IXX001,'db001.index',,DA                    IXX for DBID 1
// EXTENT SYSnnn,volser
// DLBL DEM001,'db001.dem',,DA                      Area DEM in DBID 1
// EXTENT SYSnnn,volser
// DLBL OUT1,'dembkup',n,SD                         Output data
// EXTENT SYSnnn,volser,1,0,strk,ntrks
// EXEC DBUTLTY,SIZE=750K
         BACKUP    AREA=DEM,DBID=1,DEVICE=DISK,DTFNAME=OUT1
/*
/&
* $$ EOJ
```

Sample Report: BACKUP Data Area - Native Key Sequence

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          BACKUP AREA=DEM,DBID=1,DEVICE=TAPE,DTFNAME=OUT1

                    FUNCTION=BACKUP
          ❷            AREA=DEM
                       DBID=1
                       DEVICE=TAPE
          ❸            DTFNAME=OUT1
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

      TABLE         RECORDS           OUT COMPRESSION    ROW SIZE
     ❹                                ❺
      PNC                 5           NONE                     14
      PNM                 3           NONE                     23
      POH                 5           NONE                     20
      POL                 4           NONE                     19
      TOTAL              17
```

❹

Confirmation that the request completed correctly. Seventeen records were backed up from the tables in the area. Any errors encountered during processing are noted with messages here.

❺

- OUT COMPRESSION is the format of the tables output records. It can say NONE, if the output is not compressed. It can say DB, if the output is compressed with DB compression. Otherwise, it is the name of the compression routine and the 8 characters of the compression user value.

Example JCL (Output to Tape) - Physical Sequence

The following shows the command to back up the data area ACT in database 010 to the output data set, OUT2. Output the data in physical sequence and in compressed form (default).

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL ACT010,'db010.act',,DA                     Area ACT in DBID 010
 // EXTENT SYSnnn,volser
 // ASSGN SYS010,TAPE
 // PAUSE MOUNT TAPE

 // TLBL OUT2,'act.bkup'                            OUTPUT DATA
 // EXEC DBUTLTY,SIZE=750K
    BACKUP AREA=ACT,DBID=10,DEVICE=TAPE,DTFNAME=OUT2,SEQ=PHYSICAL
 /*
 /&
 * $$ EOJ
```

Sample Report: BACKUP Data Area - Physical Sequence

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
           ❶         BACKUP AREA=ACT,DBID=10,DEVICE=TAPE,DTFNAME=OUT2,SEQ=PHYSICAL

                    FUNCTION=BACKUP
           ❷           AREA=ACT
                       DBID=10
                       DEVICE=TAPE
                       DTFNAME=OUT2
           ❸           SEQ=PHYSICAL
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

      TABLE         RECORDS           OUT COMPRESSION    ROW SIZE
     ❹                                ❺
      ACT                 3           NONE                     31

      TOTAL               3
```

❹

Confirmation that the request completed correctly. Three records were copied from the single table in the area. Any errors encountered during processing are noted with messages here.

❺

- OUT COMPRESSION is the format of the tables output records. It can say NONE, if the output is not compressed. it can say DB, if the output is compressed with DB compression. Otherwise, it is the name of the compression routine and the 8 characters of the compression user value.

Chapter 6: COMM (Communicate with Multi-User Facility)

Use the COMM function to communicate a request directly to the Multi-User Facility. It executes similar to a batch program through a task area in the Multi-User Facility. After completing an ALTER, CANCEL, CLRPXX, EOJ, NEWRXX, REQABORT, or SNAP subfunction, or the EOJOFF console only command (discussed in the *CA Datacom/DB Database and System Administration Guide*), the following message is displayed: **COMM aaaaaaaa REQUEST ACCEPTED.**

Overview

Use the COMM function when you need to perform one of the following:

- Modify the error table (only at the direction of Technical Support)
- Clear the Statistics and Diagnostics Area
- End Multi-User processing normally
- Request a new Recovery File
- Abort a long-running request
- Write a dump of the Master List to the Statistics and Diagnostics Area (PXX)
- Print/display the status of Multi-User jobs
- Other console-like commands (for details about console-like commands, see the section about maintaining CA Datacom/DB using console commands in the *CA Datacom/DB Database and System Administration Guide*).

Each option is explained in the following sections of this chapter.

Note: COMM options require that the Multi-User Facility is active.

Length of Time to Process a Request

The utility waits until the request is complete to continue.

When using the Multi-Guest Sharing Option (MGSO) and Security, COMM utility jobs must run in the Multi-User Facility guest machine.

COMM ALTER (Modify Error Table)

The COMM ALTER option modifies the error table which controls what traces CA Datacom/DB performs and what messages it generates.

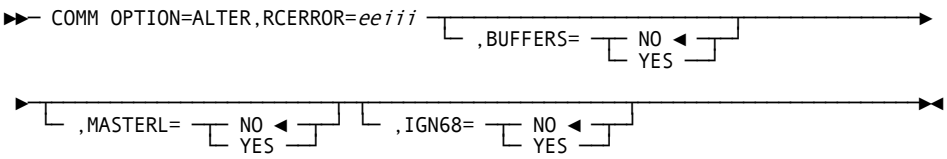
When to Use

This option should only be used to analyze processing problems at the direction of Technical Support.

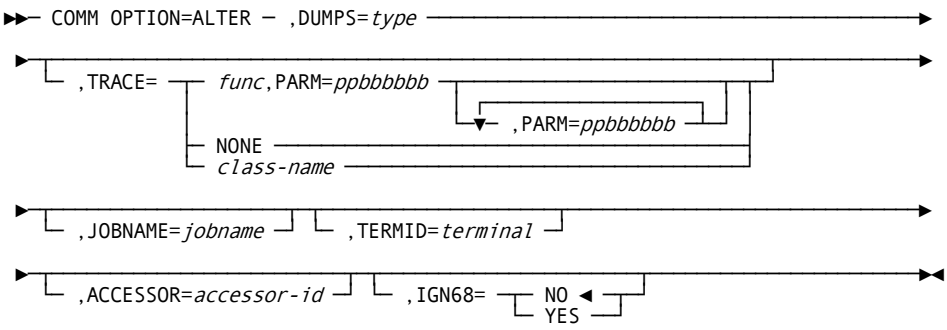
How to Use

Multi-User mode is required to execute this command. Execute the COMM function with one of the following command formats:

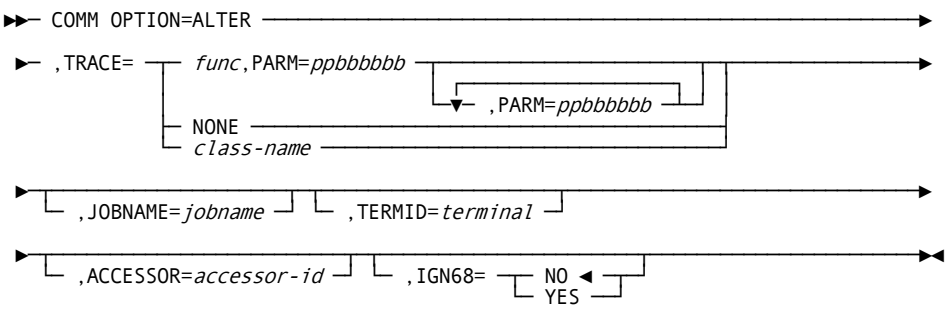
Modify Error Table, RCERROR=



Modify Error Table, DUMPS=



Modify Error Table, TRACE=



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keywords

OPTION=ALTER

Requests a change in the type of dump produced for a specific error or under specific conditions.

Optional Keywords

,BUFFERS=

(Use only when RCERROR= is specified.) Specifies if a dump of all native CA Datacom/DB buffers is to be produced.

Valid Entries:

NO or YES

Default Value:

NO

,DUMPS=

(Do not use when RCERROR= is specified.) Specifies type of dump to be produced.

ARRAY

Dump of the SQLDA, row, and input statement for each logical unit of work (LUW) in the task array

CSTACK

Dump of the C program stack

CURSOR

Dump of the cursor list and cursor pointed to by the logical unit of work (LUW)

DSFUSR

Dump of the DSFUSR array

GLOBALA

Dump of the global area

GLOBALL

Dump of the global plan list and plans and the logical unit of work (LUW), User Requirements Table list and the User Requirements Tables pointed to by each attach

GLOBALX

Dump of the X_COMM space

LUW

Dump of the logical unit of work (LUW)

PLAN

Dump of plan list and plans pointed to by the logical unit of work (LUW)

QCB

Dump of the QCB area pointed to by the logical unit of work (LUW)

SQLFLOW

A trace showing the current program name and the program name which called the current program

STMT

Dump of statement list, short statement, statement source and the QCB area

TASK

Dump of the task, the SQLDA, row and the input statement

URT

Dump of the User Requirements Table list and User Requirements Tables pointed to by the logical unit of work (LUW)

Default Value:

(No default)

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled. For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

JOBNAME=

(Use only when DUMPS= or TRACE= is specified.) Specifies the job name being traced.

Valid Entries:

1- to 8-character job name

Default Value:

All jobs

,MASTERL=

(Use only when RCERROR= is specified.) Specifies if a Master List dump is to be produced.

Valid Entries:

NO or YES

Default Value:

NO

,RCERROR=*eeiii*

(Do not use when DUMPS= or TRACE= is specified.) Specifies the return code to modify, RCERROR=**eeiii**, where **ee** is the two-character external return code and **iii** is the three-character internal return code. With the RCERROR Multi-User startup option, you can specify values for *ee* and *iii.* For details, see the *CA Datacom/DB Database and System Administration Guide*.

,TERMID=

(Use only when DUMPS= or TRACE= is specified.) Specifies the terminal ID of the terminal which is issuing the SQL request.

You can include this parameter once per COMM request.

Valid Entries:

4- to 8-character terminal ID

Default Value:

All terminals

,ACCESSOR=

(Use only when DUMPS= or TRACE= is specified.) Specifies the security signon name.

You can include this parameter once per COMM request.

Valid Entries:

1- to 8-character accessor ID

Default Value:

All accessors

,TRACE=

(Do not use when RCERROR= is specified.) Specifies what is to be traced to assist in the resolution of problems involving SQL processing. TRACE= can be used only once in a request. You can use the TRACE= parameter with or without the DUMPS= parameter.

Because of the large amount of output which can be generated by a trace, we recommend that you limit the trace by also specifying either JOBNAME= or TERMID=.

func*,PARM=*ppbbbbbb

Specifies the name of the SQL processor function for which the parameters are to be dumped, followed by one or more PARM= keywords, each with a value in the form ppbbbbbb:

pp is 01 to 99
Position in the parameter list of the parameter to be dumped.

bbbbbb is 000000 or 000001 through 999999
000000 dumps the parameter value. 000001 through 999999 dumps *bbbbbb* bytes of the area to which the parameter points.

Technical Support provides you the function and parameter value to code.

Default Value:

NONE. Turns off all trace requests.

,TRACE=

class-name

Specify an SQL trace class by replacing *class-name* with one of the following:

- TRACECONSTR Constraint
- TRACECRS Row-level activity within SQL and a summary level Compound Boolean Selection Diagnostic Report. See Reporting on the Compound Boolean Selection Facility (see page 594). This report is only generated once per statement execution, so for an inner join table that generates a Compound Boolean Selection SELFR request for each outer row, the Compound Boolean Selection Diagnostic Report is only generated for the first request. The remaining reports are usually identical. To generate a detailed Compound Boolean Selection Diagnostic Report, also turn on TRACEDETAIL.
- TRACEDD CA Datacom/DB transactions issued to process SQL requests
- TRACEDETAIL Causes more information to be traced for some other trace classes. With TRACEDETAIL:

  TRACEEXPR Shows a BEFORE execution image of expression nodes in addition to the AFTER execution image.

  TRACEMSG Turns on SQL PLAN option MSG=DD. This generates per-execution messages written to both SYSADM.SYSMSG table and the Statistics and Diagnostics Area (PXX). Also, the summary level Compound Boolean Selection Diagnostic Report is generated.

  TRACECRS Generates a detailed Compound Boolean Selection Diagnostic Report showing each index and row accessed and some information that is for internal use only. TRACECRS can generate a very large report. Turn it on only as necessary.

  TRACEDD Column and statement definitions read from DDD (Data Definition Directory).

  TRACEPARSE Dumps expression nodes as they are generated, in addition to the summary dump at the end of parsing.

  TRACECONSTR Debugging information for confirmation, creating and enforcing foreign keys and domain constraints, memory usage, and date arithmetic.

  To turn on the SQL PLAN option MSG=DD (detail messages), issue a second COMM ALTER option, specifying TRACE=TRACEDETAIL. Turning on the MSG=DD option also turns on the Compound Boolean Selection diagnostic detail reporting showing each call to the index/read module. For example, to trace detail optimization messages and produce a Compound Boolean Selection Diagnostic Detail Report for a particular job, enter the following:

  ```
  COMM OPTION=ALTER,TRACE=TRACEMSG,JOBNAME=jobname
  COMM OPTION=ALTER,TRACE=TRACEDETAIL,JOBNAME=jobname
  ```

- TRACEERR Calls to Multi-User Facility
- TRACEEXPR Expression and data type
- TRACEMEMORY Dynamic memory activity
- TRACEMSG Turns on the SQL PLAN option MSG=SS (summary messages) and Compound Boolean Selection Diagnostic Report. (See Reporting on the Compound Boolean Selection Facility (see page 594).)
- TRACEPARSE SQL parser
- TRACEPLAN PLAN locking
- PROCTRIGPLS Diagnostic information related to procedures and triggers.

  Note: PROCTRIGPLS does *not* have a prefix of TRACE.
- TRACERAAT Record-at-a-time calls issued to process SQL requests

Default Value:

(No default)

Example JCL

The following shows the command to issue an ALTER option.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
    COMM OPTION=ALTER,RCERROR=05034,MASTERL=YES,BUFFERS=YES
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM ALTER

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             COMM OPTION=ALTER,RCERROR=05034,MASTERL=YES,BUFFERS=YES

                    FUNCTION=COMM
                       BUFFERS=YES
       ❷               MASTERL=YES
                       OPTION=ALTER
                       RCERROR=05034


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

DB13105I - COMM     ALTER    REQUEST ACCEPTED     ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue.

COMM CLOSE (Close Database or Area)

You can close an open database (or all open databases) using COMM CLOSE, if no User Requirements Tables are open in the database(s) to be closed.

Beginning with Version 11.0, the COMM CLOSE DBUTLTY function (*not* the console command) can close a single area when you specify the AREA= and DBID= keywords.

Note: A database defined as virtual cannot be closed.

When to Use

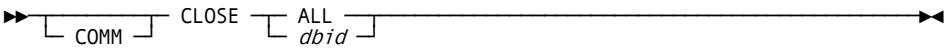
COMM CLOSE used for databases is typically used as an early step in a job stream for reorganization.

Using the DBUTLTY COMM CLOSE function to close an area would be useful if you needed to uncatalog an area, scratch an area, or extend it through DBUTLTY.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

COMM

You can omit the COMM function name when submitting this console command.

CLOSE

Requests database closure.

Required Keywords

ALL

Specify ALL, if you want all databases to be processed. CLOSE ALL closes databases that can be closed. If there are any virtual databases or databases with open URTs, a warning message is issued (for example, DB00612W COMM CLOSE AGAINST OPEN BASE(S) WITH USERS).

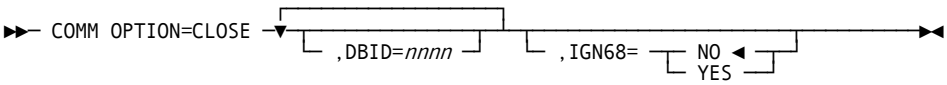
dbid

Specify a database ID, if you want only one particular database to be processed. If you request a CLOSE for a database that has an open URT, a warning message is issued (for example, DB00612W COMM CLOSE AGAINST OPEN BASE 1 WITH USERS).

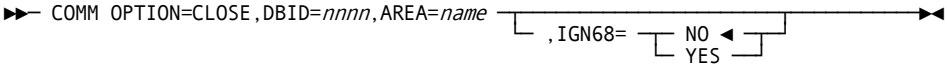
How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute the COMM function with the following command format:

Close a Full Database



Close an Area in a Database



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keywords

OPTION=CLOSE

Requests database or area closure.

,AREA=*name*

Specifying AREA=*name* lets you close a single area. If you specify the AREA= keyword you must also specify the DBID= keyword.

Use the AREA= keyword when you need to initialize, load, or extend it through DBUTLTY, and the database is currently open.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

Valid area name

Default Value:

(No default)

,DBID=*nnnn*

At the area level, you must specify a single database ID.

Valid Entries:

One valid database ID

Default Value:

(No default)

Optional Keyword

,DBID=*nnnn*

At the database level, DBID= is optional and may repeat. Each occurrence may specify one or more valid database IDs or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250).

If DBID= is not specified, all databases are processed.

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database in a range not defined is ignored.

If you request the closure of a particular database and that database is virtual or has an open URT, the command fails with a return code 46(1). If you request (by not specifying a particular DBID) the closure of all open databases, all that can be closed are closed. In addition to any closing, if any are virtual databases a warning message is issued, and if any are not virtual but have open URTs, a return code 46(1) is issued.

Valid Entries:

One or more valid database IDs or a range of DBIDs as already described.

Default Value:

ALL

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to close database DBID=1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          COMM      OPTION=CLOSE,DBID=1
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM CLOSE

Here is a sample COMM CLOSE report.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          COMM OPTION=CLOSE,DBID=1

                    FUNCTION=COMM
          ❷            DBID=00001
                       OPTION=CLOSE


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - COMM     CLOSE    REQUEST ACCEPTED     ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue.

COMM CLRPXX (Clear Statistics and Diagnostics Area (PXX))

The COMM CLRPXX option clears the Statistics and Diagnostics Area (PXX) of all data. This option has the same effect as disabling and enabling the Multi-User Facility. That is, CA Datacom/DB begins to write data to logical block one, and all data previously in the Statistics and Diagnostics Area (PXX) is lost.

Note: Summary statistics are maintained in the Multi-User Facility and are not cleared nor reset in memory.

The COMM CLRPXX function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

When to Use

Use the COMM CLRPXX option when your Statistics and Diagnostics Area is nearing capacity and you want to clear it out.

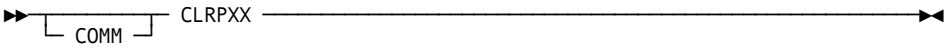
Reporting PXX Data

Be sure to run the REPORT PXX function before running CLRPXX, since all data previously recorded is lost. For details see Statistics and Diagnostics Area (PXX) Report (see page 562).

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

COMM

You can omit the COMM function name when submitting this console command.

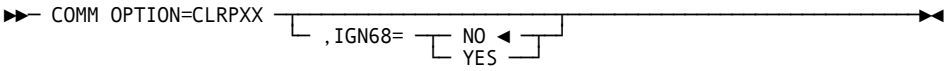
Required Keywords

CLRPXX

Requests that the Statistics and Diagnostics Area (PXX) be cleared.

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keyword

OPTION=CLRPXX

Requests that the Statistics and Diagnostics Area (PXX) be cleared.

Optional Keyword

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to clear the Statistics and Diagnostics Area (PXX).

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
           COMM     OPTION=CLRPXX
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM CLRPXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          COMM OPTION=CLRPXX

                    FUNCTION=COMM
          ❷            OPTION=CLRPXX


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - COMM     CLRPXX   REQUEST ACCEPTED     ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue.

COMM CONSOLE (Simulate Console Command)

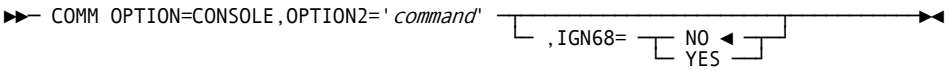
The COMM CONSOLE option provides the ability for a DBUTLTY batch execution to support a function that executes any of the console-like commands being issued to a given Multi-User Facility. For details about console-like commands, see the section about maintaining CA Datacom/DB using console commands in the *CA Datacom/DB Database and System Administration Guide*.

When to Use

Use this option as needed to issue commands with batch DBUTLTY jobs. This allows console-like commands to be issued from DBUTLTY SYSIN without the need for z/VSE level authority to issue console commands or asking the operator to issue the command on your behalf. No special setup is required to be able to use the option. Because DBUTLTY can be secured using external security packages, using this command does not result in any security exposure.

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keyword

OPTION=CONSOLE

Requests that the value for *command* specified with the OPTION2= keyword be sent to the Multi-User Facility.

,OPTION2='*command*'

Used to specify the command that is to be sent to the Multi-User Facility. DBUTLTY passes the character string provided by OPTION2= to the Multi-User Facility the same way a console modify would pass the command. It then prints a provided message as a result. Because a blank is required between the command and the value, the entire string must be placed inside single quote marks ('). If the command is accepted, message DB00608I is issued. If the command is not accepted, message DB00604I is issued. For the supported values for *command*, see the section about maintaining CA Datacom/DB with console-like commands in the *CA Datacom/DB Database and System Administration Guide*.

Optional Keyword

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to simulate a console command:

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          COMM     OPTION=CONSOLE,OPTION2='command'
 /*
 /&
 * $$ EOJ
```

COMM EOJ (End Multi-User Facility Processing)

The Comm EOJ function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

When to Use

Use the COMM EOJ option to end the Multi-User Facility in an orderly fashion. This is the recommended way to terminate the Multi-User Facility.

Processing of EOJ Requests

The COMM EOJ option informs the Multi-User Facility that processing is to end. After receiving an EOJ request, the Multi-User Facility does not allow any new jobs to start processing. Multi-User allows jobs in progress to continue. The Multi-User Facility goes to EOJ when the last job completes. A job is considered complete when all open User Requirements Tables have been closed. When the Multi-User Facility is terminated an EOJ report is written to the SYSLST file. The report contains information about:

- An echo of Multi-User startup options
- Memory Resident Data Facility Summary Information
- Accounting Facility Summary Information
- TCB Use Summary Information
- Miscellaneous Diagnostic Information
- Also provided, as shown in EOJ Reports (see page 151), are the following:
  - Procedure TCB Use Summary Information
  - Procedure Summary Information
  - Return Code Summary Information
  - SQL Code Summary Information
  - Memory Use Summary Information
  - MUF EOJ Log Status Information
  - Other Information

  In addition, be aware of the following:

  - The Log Area (LXX) Output report (see Sample Report: REPORT LXX)
  - The Detail Report for Reprocessed Requests (in the *CA Datacom/DB Database and System Administration Guide*, see the chapter on restarting the Multi-User Facility)
  - The Detail Report for Active Transactions (in the *CA Datacom/DB Database and System Administration Guide*, see the chapter on restarting the Multi-User Facility)

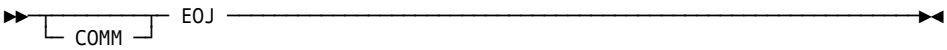
No other COMM function, except the COMM EOJOFF console command, is accepted by the Multi-User Facility after this option is initiated. Once an EOJ request is accepted by the Multi-User Facility, the request can *only* be canceled with the COMM EOJOFF command. Unless EOJOFF is used, the Multi-User Facility must be allowed to complete termination processing. See the *CA Datacom/DB Database and System Administration Guide* for details about using EOJOFF. After the Multi-User Facility is started again, new jobs can be executed.

Online systems must initiate a close request for each User Requirements Table opened.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

COMM

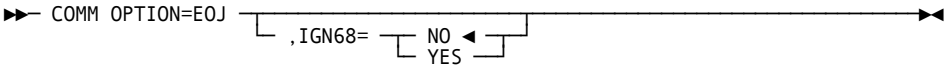
You can omit the COMM function name when submitting this console command.

EOJ

Indicates the EOJ option of the COMM function is the desired action.

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute the COMM function with the following command format.



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keyword

OPTION=EOJ

Requests normal termination of the Multi-User Facility.

Optional Keyword

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the Multi-User Facility (MUF) connection can wait if the MUF is not currently available. But if IGN68=YES is specified, DELAY68= is ignored and the function continues. If the delay is desired, however, either do not specify IGN68=, in which case it defaults to NO, or specify IGN68=NO.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to issue the EOJ option.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
           COMM     OPTION=EOJ
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM EOJ

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          COMM OPTION=EOJ

                    FUNCTION=COMM
          ❷            OPTION=EOJ


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - COMM     EOJ      REQUEST ACCEPTED    ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue.

Summary Reports Produced

This section shows examples of the following:

- Multi-User Startup Options Echoed
- Memory Resident Data Facility Summary Information
- Accounting Facility Summary Information
- TCB Use Summary Information
- Miscellaneous Diagnostic Information Report

See EOJ Reports (see page 151) for these additional summary reports:

- Procedure TCB Use Summary Information
- Procedure Summary Information
- Return Code Summary Information
- SQL Code Summary Information
- Memory Use Summary Information
- MUF EOJ Log Status Information
- Other Information
- Task Information

Multi-User Startup Options Echoed

When the Multi-User Facility is terminated, a list of the Multi-User startup options specified is written to SYSLST. For information on the Multi-User startup options, see the *CA Datacom/DB Database and System Administration Guide*.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

DB01900I - *ACCT   0,4K,OPEN,4K              ACCT'ING - TABLE, BUFFER, STS, THRH  }
DB01900I - *ACCT   1,8K,CLOSED,2K            ACCT'ING - TABLE, BUFFER, STS, THRH  }
DB01900I - *ACCT   2,16K,OPEN,6K             ACCT'ING - TABLE, BUFFER, STS, THRH  }
DB01900I - *ACCT   3,32K,CLOSED,1K           ACCT'ING - TABLE, BUFFER, STS, THRH  }
DB01900I - *ACCT   4,64K,OPEN,32K            ACCT'ING - TABLE, BUFFER, STS, THRH  }
DB01900I - *ACCT   5,128K,CLOSED             ACCT'ING - TABLE, BUFFER, STS, THRH  }
DB01900I - *ACCTPRM 4                        ACCT'ING - DBID OF PRM TABLE         }
DB01900I - BREAK   10                        ALLOW TASK SWITCHING                 }
DB01900I - CBS     6,32K,0,0                 CBS - DBID, BUFFER, STEN, STIO       }
DB01900I - COVERED 1                         ALL OF BASE 1 (FIRST)                }
DB01900I - COVERED IXX999,80K,ACTIVE         PART OF INDEX                        }
DB01900I - COVERED DEM999,40K,ACTIVE         PART OF DATA                         }
DB01900I - DATACOM DB,SQL,DTCMSRVR                                                }
DB01900I - DATAPOOL 8192,40                  DATALN,DATANO                        }
DB01900I - DICTIONARY 2,15                   DATA-DICT DBID,DDD DBID              }
DB01900I - EXCTLNO 16                        EXCLUSIVE CONTROL MAX (IGN IF SQL)   }
DB01900I - EXPAND  4096,4                    EXPAND BUFFERS - LENGTH, NUMBER      }
DB01900I - *FLEXPOOL 2,2,2                                                        }
DB01900I - *IOTASK  2                        IOTASK MAX/CURR,CURR,DSP             }  ❶
DB01900I - LOGPEND 100                       LOGGING PEND NUMBER                  }
DB01900I - LOGPOOL 3                         LOG TXB BUFFERS                      }
DB01900I - LOGRCV  NEVER                     LOGGING RECOVERY                     }
DB01900I - LOGSPILL 99                       SPL2,SPL3,SPL1                       }
DB01900I - MAXELRQ 255                       MAXIMUM ELEMENT PER REQUEST          }
DB01900I - MUF DBDVMUF                       MUF NAME FOR DYNAMIC TABLES          }
DB01900I - NONSWAP YES                       MAKE IS NON-SWAPPABLE IF AUTHORIZED  }
DB01900I - PXXSTATS EOJ                      PXX STATS (DETAIL,EOJ,EOJPRT,NONE)   }
DB01900I - *SMPTASK  1                                                            }
DB01900I - *SMPTASK  1                        IOTASK MAX/CURR,CURR,DSP            }
DB01900I - *SQLDEFAULT 16,CASQLDEFAULT        AREABASE,AREANAME                   }
DB01900I - *SQLOPTION YES,17,DATACOM,120      SQL,TTMID,SQLMODE,LUWTOUT           }
DB01900I - SUBTASK NONE                      NO SUBTASK                           }
DB01900I - SYSPOOL 4,10,20                   CXXNO,IXXNO,DXXNO
DB01900I - TASKS   6,8K                      USER TASKS,SIZE,VMTASK,VAECOM
DB01900I - VIRTUAL 6,2M                      CBS VIRTUAL
```

❶

Echo of the Multi-User startup options.

Memory Resident Data Facility Summary Information

When the Multi-User Facility is terminated, summary Memory Resident Data Facility (MRDF) information is written to the SYSLST file after the Multi-User startup options.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                                        M R D F     SUMMARY INFORMATION


  ❷        ❸

                                                                 VIRTUAL        COVERED        COVERED
AREA      USE      SIZE      BLOCKS  SPACE      MRDF READS   MRDF WRITES  1ST/ACT READS  NON-1ST READS     PERCENT

IXX00001  COVERED   83K          22  ADDRESS             1                            4              0  WITHIN 100
DEM00001  COVERED   62K          62  ADDRESS             1                            2              0  WITHIN 100
PAY00001  COVERED   62K          62  ADDRESS             1                            2              0  WITHIN 100
IXX00006  VIRTUAL    2M         510  ADDRESS             6             3                                 USED   0
IXX00999  COVERED   80K          21  ADDRESS             1                            3                 REUSED  25
DEM00999  COVERED   40K          40  ADDRESS             1                            2                 REUSED  33
```

Note: The spacing in this example is expanded for clarity.

❷

AREA - Area DATACOM-NAME and database DATACOM-ID in the format aaannnnn.

❸

USE - Indicates whether the resident area is COVERED or VIRTUAL.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                                        M R D F     SUMMARY INFORMATION


                    ❶          ❷        ❸          ❹          ❺             ❻            ❼          ❽

                                                                 VIRTUAL        COVERED        COVERED
AREA      USE      SIZE      BLOCKS  SPACE      MRDF READS   MRDF WRITES  1ST/ACT READS  NON-1ST READS     PERCENT

IXX00001  COVERED   83K          22  ADDRESS             1                            4              0  WITHIN 100
DEM00001  COVERED   62K          62  ADDRESS             1                            2              0  WITHIN 100
PAY00001  COVERED   62K          62  ADDRESS             1                            2              0  WITHIN 100
IXX00006  VIRTUAL    2M         510  ADDRESS             6             3                                 USED   0
IXX00999  COVERED   80K          21  ADDRESS             1                            3                 REUSED  25
DEM00999  COVERED   40K          40  ADDRESS             1                            2                 REUSED  33
```

Note: The spacing in this example is expanded for clarity.

❶

SIZE - Amount of memory allocated for this area that contains the data, Index, or Log Area blocks. This is the amount used, not the size specified in the Multi-User startup options. SIZE values are printed as follows:

- Numbers 1 through 9999 are printed with a blank terminator, for example, 9999 is printed as 9999 followed by a blank.
- Numbers 10000 through 9999K are printed as n through nnnnK, rounded up.
- Numbers 9999K plus one through 9999M are printed as n through nnnnM, rounded up.
- Numbers above 9999M are printed with a G as the terminating character, rounded up.

❷

BLOCKS - The number of data, Index, or Log Area blocks allocated to memory. COVERED is allocated in blocks and VIRTUAL is allocated in tracks.

❸

SPACE - ADDRESS indicates use of normal partition memory.

❹

MRDF READS - Provides the number of blocks that were read from the MRDF space instead of from disk.

❺

VIRTUAL MRDF WRITES - Provides the number of blocks that were written to the MRDF VIRTUAL space.

❻

COVERED 1ST/ACT READS - Provides the number of blocks read from disk for either a FIRST or ACTIVE COVERED area.

❼

COVERED NON-1ST READS - For a COVERED FIRST area only, provides the number of blocks read from disk which do not reside in the MRDF buffers.

❽

PERCENT - Indicates the amount of use based on the following:

REUSED

(COVERED ACTIVE areas only.) Provides the percent of MRDF reads out of the total reads.

USED

(Virtual Areas Only.) Provides the percent of the space allocated which was actually used to service program requests.

WITHIN

(COVERED FIRST areas only.) Provides the percent of the total reads which occurred within the COVERED FIRST buffers.

Accounting Facility Summary Information

If the Accounting Facility is generated in this Multi-User Facility and not disabled, an Accounting report follows the MRDF report.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                        ACCOUNTING SUMMARY INFORMATION


TABLE NAME:  A04                      DATABASE:  5
REQUESTS PROCESSED:  7                EXCPS:  14
LOCATES:  6                           ADDS:  0                                UPDATES:  3
ENTRY UNAVAILABLE:  0                 EXCLUSIVE CONTROL CONFLICTS:  0         SKIPPED DUE TO EXCLUSIVE:  0
CORE SIZE:  65,536                    NUMBER OF ENTRIES:  613                 ENTRIES RECLAIMED:  0
LOCATES INITIATED:  1                 SPILLS INITIATED TO AXXTHLD:  0         INTERNALLY INITIATED SPILLS:  0
```

ADDS

Number of new records added to the accounting table.

Tuning Information:
To decrease this number, decrease the complexity of the key.

CORE SIZE

The buffer parameter specified for this table.

DATABASE

Database ID containing the accounting table.

ENTRIES RECLAIMED

Number of in-memory table entries reused after a spill, that is, the number of entries for which a LOCATE was not required.

Tuning Information:
To increase this number, increase the buffer parameter, decrease the complexity of the key, or both.

ENTRY UNAVAILABLE

Number of user application requests that had to wait due to insufficient memory allocated by the buffer parameter.

Tuning Information:
To decrease this critical value, decrease the complexity of the key, increase the value of the buffer parameter, or both.

EXCLUSIVE CONTROL CONFLICTS

Number of entries in the accounting table that had exclusive control held by a user application when a spill was initiated.

EXCPS

Number of execute channel programs (EXCPs) issued for this accounting table.

Tuning Information:
This value is adversely affected by:

- The number of records processed.
- The number of spills initiated due to the threshold parameter of the Multi-User startup option ACCT being exceeded.
- The value of the buffer parameter of the ACCT option being too small.

INTERNALLY INITIATED SPILLS

Number of spills initiated to decrease the number of entries unavailable.

Tuning Information:
To decrease this value, increase the buffer parameter value, decrease the complexity of the key, or both.

LOCATES

Number of entries retrieved into the accounting buffers including existing entries in the accounting table with reclaimed internal space.

Tuning Information:
To minimize this number, decrease the complexity of the key, if the number of adds is high, or increase the size of the buffer parameter, if the number of updates is high (see UPDATES below).

LOCATES INITIATED

Number of times that a group of locates were performed for the accounting table. This parameter is not applicable to tuning and is for use by Technical Support.

NUMBER OF ENTRIES

Number of entries in the in-memory accounting buffer (always odd).

REQUESTS PROCESSED

Number of times this accounting table was used. This value is determined by the conditional expression, if there is one.

SKIPPED DUE TO EXCLUSIVE

Number of entries that were not immediately processed due to exclusive control conflicts.

SPILLS INITIATED TO AXXTHLD

Number of times that the threshold parameter value of the Multi-User startup option ACCT was exceeded. (This value is reset after each internally initiated spill.)

Tuning Information:
To decrease this value, increase the threshold parameter value.

TABLE NAME

Name of the accounting table.

UPDATES

Number of times that records in the accounting table were updated.

Tuning Information:
To decrease this value, increase the buffer and threshold parameters, decrease the complexity of the key or perform all these.

Miscellaneous Diagnostic Information Report

The Miscellaneous Diagnostic Information Report can be useful in dealing with Technical Support and in some tuning. Similar to the Internal Statistics dynamic system table, it provides statistics for the entire Multi-User Facility run. This report is available to SQL and non-SQL sites. The first section of the report provides multi-block read information. Multi-block reads can be done during sequential commands GETIT and GETPS based upon the User Requirements Table option SEQBUFS= and the contents of the actual table(s) being read. It is also done during some ADDITs when using the *next overflow* blocks. Multi-block read is also done (under some conditions) during Compound Boolean Selection processing. This section of the report provides the number of times various numbers of DASD blocks were read with a single EXCP. Some control exists for GETIT and GETPS, and this information is provided for information and possible tuning.

The next section of the report provides multi-block write information. With pipelining, it is possible to have multiple blocks in an area that need to be written. When possible, CA Datacom/DB writes them together with one EXCP. This usually speeds I/O processing. These counts are provided for information and cannot be tuned.

The last section reflects miscellaneous information about the Multi-User execution. The sample shows one counter in this last section, BREAKS DONE. For an explanation of BREAKS DONE and other counters that could appear (if they are non-zero), see the table that follows the sample.

Note: Items in the report which have a zero count are skipped.

Following is the sample report.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                        MISCELLANEOUS DIAGNOSTIC INFORMATION
      EVENT                    COUNT

      WRITE BLOCKS  1              6
      WRITE BLOCKS  2             11
      WRITE BLOCKS  3              2
      WRITE BLOCKS  4              9
      WRITE BLOCKS  5              3
      WRITE BLOCKS  6             10
      WRITE BLOCKS 13              1
      WRITE BLOCKS 14              1
      WRITE BLOCKS 15              1
      WRITE BLOCKS 17              4
      WRITE BLOCKS 18              1
      WRITE BLOCKS 19              5
      WRITE BLOCKS 20              5
      WRITE BLOCKS 21              2
      WRITE BLOCKS 32              1
      WRITE BLOCKS 36              1
      WRITE BLOCKS 39              5
      WRITE BLOCKS 41              1
      WRITE BLOCKS 44              1
      WRITE BLOCKS 46              5
      WRITE BLOCKS 48             28
      BREAKS DONE              1,189
```

The following explains the possible counters that could appear in the last section of the previously shown report, if those counters are not zero (counters with a value of zero are not shown in the report). The only counter displayed in the previously shown report is BREAKS DONE.

BREAKS DONE

If any break was done (in support of the BREAK Multi-User startup option), it is noted here. A break represents a time when a lower priority task which is looking at many index and or data buffers can release control and allow any waiting higher (or equal) priority task to run.

IXX QUEUE DBIDS

The maximum number of DBIDs in the index queue.

IXX QUEUE DEPTH

The maximum depth (that is, maximum number of blocks) of the index queue.

LOG VAR MAX GT8

This count is printed if you have allocated more than 8 buffers. It tells you the maximum number of buffers that have actually been used. If printed, it cannot be less than 9 nor more than the number of log-ahead buffers. It represents a high-water mark to buffer usage because the detail used counts stop at 8.

LOG VAR NO AVL

This is the count of times all log-ahead buffers are full and a wait needs to occur for a current block to free one or more buffers. If this count is large, you might want to increase the number of log-ahead buffers.

LOG VAR USED 1

This count is the number of times the first log-ahead buffer is activated. It occurs only when the single current buffer is full and no log-ahead buffers are in use. If activated, the buffer may contain only one record or a full block of records before it is written. A similar count is provided for blocks 2-7. A special count is provided for block 8, which is printed as 8+ because users can define more than 8 log-ahead buffers, and the 8 count is the number of times the 8th or 9th to the end are used (similar to the 5+ buffer reuse counters).

LOG VAR WRITE 2

This count is the number of times the current log block was written with a chain of multiple blocks. These statistics are printed in the Muf EOJ report if they are not zero. The first counter provides for the times that two log current blocks are written with a single I/O, reported as LOG VAR WRITE 2. Additional counts are provided for writing three blocks, four blocks, five blocks or six blocks. In general two blocks per track will exist and so this represents 3 tracks of log data.

IO WAIT CXX BFR

Count of number of times a task waited on a CXX I/O that was already in process.

IO WAIT IXX BFR

Count of number of times a task waited on an IXX I/O that was already in process.

IO WAIT DXX BFR

Count of number of times a task waited on an DXX I/O that was already in process.

IO WAIT DATA BFR

Count of number of times a task waited on an Data I/O that was already in process.

IO WAIT LXX BFR

Count of number of times a task waited on an LXX I/O that was already in process.

LOCK LXX CUR RC

A task is waiting on access to current log buffer.

LOCK WAIT CXX B

A task is waiting on a specific Directory (CXX) block to become available.

LOCK WAIT LRU C

A task is waiting on the least recently used chain for a Directory (CXX) buffer.

MRDF IXX DIRECT

Count of times the MRDF facility could reference an IXX block directly without moving it to an IXX buffer.

OPEN EXIT WAITS

A task is waiting on open exit.

REBINDS DONE

Count of number of automatic plan rebinds that SQL has successfully completed.

WAIT_DBIO_24BIT

Count of number of times MUF did not start an I/O because its 24-bit memory was over the size limit specified with the Multi-User startup option X_IO_24BIT_SIZE.

X_IO_LOW_THRESHOLD

The count of times CA Datacom/DB constrained and started no I/O because the available memory was below the low threshold. This can occur once per 4/100th of a second in the MUF main task and any I/O subtask attempting to start an I/O. For information about the Multi-User startup option X_IO_LOW_THRESHOLD, see the *CA Datacom/DB Database and System Administration Guide*.

X_IO_HIGH_THRESHOLD

The count of times CA Datacom/DB constrained and started no I/O because the available memory was both below the high threshold and above the low threshold, and also because 30 or more I/Os were currently outstanding. For information about the Multi-User startup options X_IO_HIGH_THRESHOLD and X_IO_LOW_THRESHOLD, see the *CA Datacom/DB Database and System Administration Guide*.

X_IO_OS_THRESHOLD

The count of times CA Datacom/DB constrained and started no I/O because the outstanding I/O was 480 or more.

EOJ Reports

This section shows examples of the following:

- Procedure TCB Use Summary Information
- Procedure Summary Information
- Return Code Summary Information
- SQL Code Summary Information
- Memory Use Summary Information
- MUF EOJ Log Status Information
- Other Information
- Task Information

Procedure TCB Use Summary Information

Following is an example of a new report on Task Control Block (TCB) use.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                        PROCEDURE TCB USE SUMMARY INFORMATION
  ❶       ❷              ❸              ❹                  ❺
         LAST          PROCEDURE                          SQL
 TCB  PROGRAM            STARTS     TCB CPU TIME      STATEMENTS

   1  B580PA01               5     000:00:00.03              5
   2 B580PA02                5     000:00:00.73              0
   3 ?                       0     000:00:00.00              0
   4  B580PA01               1     000:00:00.00              1
   5 B580PA02                1     000:00:00.14              0
   6 ?                       0     000:00:00.00              0
     7-9                     0     000:00:00.00              0
```

❶

Under the TCB heading, the number of TCBs assigned for procedures are indicated. Those that have been used at least once have a detail line. Those never used have only a range reflected under the LAST PROGRAM heading. Nested TCBs are indented by one position to note that they are different from the non-nested TCBs. Nested TCBs are only used when an SQL request is issued by the previous TCB in that nest group.

❷

Under the LAST PROGRAM heading is shown the last program to be processed in this TCB. A question mark (?) in a nested TCB is used to reflect it has never been used.

❸

Under the PROCEDURE STARTS heading is shown the number of times this particular TCB was selected to process a user procedure (caused by a call procedure or a trigger).

❹

Below the TCB CPU TIME heading are found the hours, minutes, seconds, and sub-seconds of execution time.

❺

Under the SQL STATEMENTS heading is shown the number of SQL statements executed from any program that executed in this TCB. This is zero in the last TCB of each nest group.

Procedure Summary Information

Following is an example of a Procedure Summary Information report.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                        PROCEDURE SUMMARY INFORMATION
        ❶             ❷             ❸                ❹               ❺
                     PROCEDURE                                    SQL           TCB
      PROGRAM          STARTS     TCB CPU TIME      STATEMENTS       FAILURES

      B580PA01              6     000:00:00.03              6              0
      B580PA02              6     000:00:00.87              0              0
```

❶

Under the PROGRAM heading is the name of the program that was executed.

❷

Under the PROCEDURE STARTS heading is shown the number of times this program was started for execution.

❸

Under the TCB CPU TIME heading is the time this procedure spent executing.

❹

Under the SQL STATEMENTS heading is shown the number of SQL statements executed by all occurrences of this program.

❺

Under the TCB FAILURES heading, either 'LE' was unable to fetch the module, or the module abended.

Return Code Summary Information

The following example of an EOJ Report provides Return Code Summary Information:

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                        RETURN CODE SUMMARY INFORMATION
          ❶
         RETURN CODE    OCCURRENCES    SHORT TEXT  ❷

          01 (032)        3    INVALID DB COMMAND
          03 (027)        1    KEY NAME NOT IN CXX
          05 (034)        5    TABLE HAS NOT BEEN OPENED
```

❶

Note that these are CA Datacom/DB return codes. Not all return codes, however, are listed in these counts. Several return codes that occur frequently in non-error situations are not counted. These include (but are not restricted to) return codes: 14, 19, and 04(170). Also not counted are certain return codes that occur outside the Multi-User Facility. These include (but are not restricted to) return codes: 36, 37, 57, 68, 85, and 86.

CA Datacom/DB, in its internal processing, sometimes generates return code situations that are not errors and are not externalized other than in this report or in a Master List dump, if enabled by a user using the RCERROR option. These counts cannot be clearly identified, but must be ignored as normal. They include occurrences of 12(045), 61(061), and possibly others.

❷

The SHORT TEXT is provided to speed the review of errors and save time that could be spent looking up message meanings.

SQL Code Summary Information

The following example of an EOJ Report provides SQL Codes Summary Information:

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                        SQL CODE SUMMARY INFORMATION
             SQL CODE    OCCURRENCES
                  ❶
                 -14             2
                 -20             1
                 -56             1
                -124             2
```

❶

Note that these are SQL error codes. Also note that SQL code 100 (that is, +100, not -100) is not listed in these counts because it indicates a non-error situation. Specifically, SQL code 100 indicates NO ROW FOUND and can occur during a DELETE or UPDATE, if no rows are found to delete or update. It is a normal, non-error condition for which no action is required, and therefore it is not counted in the SQL Code Summary Information report.

Memory Use Summary Information

Following is an example of an EOJ Report that provides Memory Use Summary Information. The memory usage reported is that which exists at Multi-User Facility termination. It does not include memory allocated by the operating system. It does, however, include most memory allocated by CA Datacom/DB, but not all memory.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                        MEMORY USE SUMMARY INFORMATION
          ❶         ❷                    ❸      ❹     ❺            ❻                        ❼
         DESC  LONG DESCRIPTION        24BIT-----ANY---64BIT (M=MEG,K=1024,ELSE BYTES) / OCCURRENCES

               DBMEMPR CALLS-----------   61       0     ❽
               ERROR, NON-SQL----------    1       0     ❾

         MAIN  DBMEMPR OWNED MEMORY, 24  128-K     0     ❿
         MA31  DBMEMPR OWNED MEMORY,ANY    0       2--M
         PC31  COMMON, POOL 0, ANY ----    0       8-K
         MLB4  MUF STARTUP, 24 BIT ----   29-K      0
         MLB1  MUF STARTUP, ANY -------    0     304-K
         DCB   DCB/DTF, 24 BIT --------   32-K      0
         DBIO  I/O CONTROL BLOCKS -----   32-K      0
         4K31  MISC. 4K POOL, ANY -----    0      24-K
         BDES  IXX/DXX/DATA/LXX BDES --    0      32-K
         BUFS  IXX/DXX/DATA/EXP.BFRS --    0     564-K
         DMP   PXX PRINT CACHE --------    0      33-K
         MRDF  COVERED/VIRTUAL NOT DTSP    0      47-K
         LOGB  LOG BUFFERS ------------    0     112-K
         LOGT  LOG TRACE TABLE --------    0      32-K
         DSM   DSM TRACE TABLE --------    0      32-K
         MA64  DBMEMPR OWNED MEMORY, 64    0       0        6--M
         IX64 IXX BFRS ----------------    0       0       81-K
         MRDF  COVERED/VIRTUAL 64BIT       0       0      200-K
```

❶

The first column is a short description that can be blanks and can be duplicated, if appropriate.

❷

The second column is a long description that can be blanks and can be duplicated, if appropriate.

❸

The third column is the amount of 24-bit requested memory, except for one case where it is a count.

❹

The fourth column is the amount of any requested (usually 31-bit) memory, except for one case where it is a count.

❺

The fifth column is the amount of any requested 64-bit memory.

❻

The amount is presented as follows:

- Numbers 1 through 9999 are printed with a blank terminator, for example, 9999 is printed as 9999 followed by a blank.
- Numbers 10000 through 9999K are printed as n through nnnnK, rounded up.
- Numbers 9999K plus one through 9999M are printed as n through nnnnM, rounded up.
- Numbers above 9999M are printed with a G as the terminating character, rounded up.

❼

This column lists the number of occurrences of LUW, STME, STMR, and XCL (see the descriptions in the table given in ❿ below).

❽

This line, the second line in z/OS but the first line in z/VSE, is a count of the number of memory manager requests in the 24-bit column and zero in the ANY column.

❾

There are three possible counters which can be printed here. Each can be printed if they are non-zero counts. The three possible counters are as follows:

- OUT OF STORAGE indicates the number of times a request to get memory could not be performed because no memory was available to satisfy the request.
- ERROR, NON-SQL is a parameter error on a memory manager request. This counter is used only by Technical Support. Users have no control over it.
- ERROR, SQL is a parameter error on a memory manager request. This counter is used only by Technical Support. Users have no control over it.

❿

The MAIN and MA31 lines represent the amount of memory obtained from the operating system for use as dynamic memory. MAIN contains only 24-bit memory, while MA31 is ANY requested memory, generally 31-bit. This size is never reduced, that is, once obtained from the operating system this memory is used and reused as necessary.

The lines following the MAIN and MA31 lines represent a current breakdown of where the MAIN and MA31 memory is specifically allocated. Each line represents one or more pools. Some pools can occur more than once and some of those are collapsed for reporting purposes. General descriptions of the possible pools are as follows:

DESC	LONG DESCRIPTION	Explanation
DCB	DCB/DTF, 24 BIT --------	24-bit memory used for operating system control blocks.
DBIO	I/O CONTROL BLOCKS -----	24-bit memory used during I/O.
4K31	MISC. 4K POOL, ANY -----	Dynamic 4K pool of memory used for many items such as open areas, open tables, open data sets, open extents, open CXX base/area/table information.
ACCT	ACCOUNTING BFRS --------	Accounting tables buffers.
BDES	IXX/DXX/DATA/LXX BDES --	Index, data, and log buffer directory entries.
BUFS	IXX/DXX/DATA/EXP.BFRS --	Index, data, and expand buffers.
CAP	CXX CATALOG, DDL -------	Work areas during a CXX catalog or during SQL DDL statements.
CBSS	CBS SET	CBS set buffer
CBSW	CBS Work	Working storage for CBS
CNSB	CONSTRAINT BUILD OBJECT	SQL constraint processing memory, build object.
CNSC	CONSTRAINT CHECK ADD --	SQL constraint processing memory, check during add.
CNSR	CONSTRAINT READ TO EXEC	SQL constraint processing memory, read to execute.
CNST	CONSTRAINT TEMP. STORAGE	SQL constraint processing memory, temporary storage.
DBLG	SAVED LOG RECORD ---	Transaction backout saved log record.
DMP	PXX PRINT CACHE --------	Buffer to cache SQL and CBS trace and debugging information.
DSM	DSM TRACE TABLE ---	Space management trace table.

DESC	LONG DESCRIPTION	Explanation
IX64	IXX BFRS -------	IXX buffers in 64-bit memory.
KYEL	CXX KEY/ELEMENT --------	CXX key and element memory for open tables.
LOGB	LOG BUFFERS ----------	Log buffers.
LOGT	LOG TRACE TABLE --------	Log trace/debugging table.
LUW	LOG.UNIT.WORK STORAGE --	SQL memory for logical unit of work control. Number of occurrences listed in OCCURRENCES column.
MA64	DBMEMPR OWNED MEMORY, 64	If 64-bit memory has been used by MUF, the report contains a MA64 line showing the total memory acquired in 64-bit.
MFR	RESTART WORK TABLE ---	Restart work tables and User Requirements Tables.
MLB1	MUF STARTUP, ANY -------	Done during MUF enable. Contains static control blocks allowed to be in 31-bit memory.
MLB4	MUF STARTUP, 24-BIT ----	Done during MUF enable. Contains static control blocks required to be in 24-bit memory.
MPZ	MPZ TRACE TABLE ---	*(z/OS only)* Data Sharing, MUFplex, trace table.
MRDF	COVERED/VIRTUAL NOT DTSP	Control information for Covered/Virtual usage required in the MUF address space. Also contains buffers when running z/OS and not running Authorized.
MRDF	COVERED/VIRTUAL 64BIT	MRDF in 64-bit memory
MSC	MISCELLANEOUS SQL ------	SQL miscellaneous memory.
NEWC	NEW COPY WORK AREA ---	Work area for COMM NEWCOPY.
PC31	COMMON, POOL 0, ANY ----	Miscellaneous small uses of 31-bit memory.
PCOM	COMMON, POOL 0, 24-BIT -	Miscellaneous small uses of 24-bit memory.
PLN	PLAN STORAGE ----------	SQL plan memory.
PRCD	PROCEDURE RELATED ------	SQL procedure-related storage.
PRS	PARSER TEMP. STORAGE ----	SQL parser memory.
REQ	REQUEST SQL STORAGE ----	SQL request memory.
SASV	STXIT WORK AREAS -------	z/VSE program failure work areas.
SCF1	SECURITY FAST PATH-USER	Security fast path for users.
SCF2	SECURITY FAST PATH-TABLE	Security fast path for tables.
SPILL	LOG SPILL BUFFERS	Buffers used for spilling the log to the RXX.

DESC	LONG DESCRIPTION	Explanation
STMC	STMT PREPARE: CURSOR UPDSQL	SQL statement prepare - cursor update.
STMD	STMT EXECUTE: DYN-CURSOR	SQL statement execution - dynamic cursor.
STME	STMT EXECUTE: STMT COPY	SQL statement execution cache. Number of occurrences listed in OCCURRENCES column.
STMN	STMT EXECUTE: NON-STATIC	SQL statement execution - non-static.
STMP	STMT PREPARE: GENERAL --	SQL statement prepare - general use.
STMR	STMT READ PROCESSING ---	SQL statement read processing. Number of occurrences listed in OCCURRENCES column.
STMS	STMT SEARCH: DB2 MODE --	*(z/OS only)* SQL statement search - DB2 mode statements.
TSKW	TASK (RWTSA) WORK AREAS	Work areas associated with each task area being used.
VEW	VIEW RESOLUTION --------	SQL view resolution.
VPEP	DD VPE GLOBAL ----------	Datadictionary global work area.
XCF	MVS XCF USAGE ----------	Contains the work areas required for XCF communication in z/OS. This occurs when using XCF to run jobs on an image other than the one on which the Multi-User Facility is executing or during Data Sharing.
XCL	SQL C STACK, TASK WORK -	Work areas for SQL task processing. Number of occurrences listed in OCCURRENCES column.
XES1	DATASHARING (MUFPLEX)	*(z/OS only)* Data Sharing, MUFplex, trace/debugging table.

MUF EOJ Log Status Information

The following is an example of an EOJ Report that you get if the MUF opened the log.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


                                        MUF EOJ LOG STATUS INFORMATION

MUF EOJ, LOG CLOSED, CLEAN EOJ                  ❶
LOG CONTROL BYTES-229,376 ROLLBACK-3,963        ❷
LOG RECORD BYTES-22,373,107 WRITTEN-24,891,580  ❸
LOG RECORD SEQ-3 LOG BLOCK SEQ-243              ❹
```

❶

This message indicates that there was a *clean* EOJ. If the EOJ had not been clean, other messages would have appeared here.

❷

This line gives LOG CONTROL BYTES information, that is, the number of bytes written when writing the control block. The control block is written as required. It is possible to use X_LOG_CONTROL_BLKS to control this writing, and if you are changing the defaults, the count shown in LOG CONTROL BYTES shows the variations. However, while changing the values are possible, we do not recommend your doing so. More writes allow a faster restart process. Fewer writes lengthen the restart process.

On the same line is ROLLBACK bytes written information. This refers to the rewritten blocks that have had one or more log records subject to rollback (transaction backout) and that therefore needed to be rewritten. This count indicates, therefore, the log records that have been undone. You have a certain amount of control over this process when your applications are designed to do frequent ROLBK or LOGTB commands.

❸

This line gives LOG RECORD BYTES n, where n can be up to 31 digits, the total number of bytes in all the log records built by this MUF. The number is not affected by how CA Datacom/DB chooses to write the bytes to DASD. The number is not affected by spilling or transaction backout.

With LOG RECORD BYTES, there is also WRITTEN information displayed. WRITTEN n information, where n is up to 31 digits and indicates the total number of bytes written to local DASD.

With some logging options, WRITTEN n and LOG RECORD BYTES n match, but with other logging options they do not match. To maximize bytes written to a given track, CA Datacom/DB rewrites bytes.

❹

This line gives the Log Record Sequence Number, that is, the count of log records added during this MUF execution, except that the count can be reset to 1 during QUIESCE processing. In a MUFplex environment, the MUF execution is defined as the enabling of a MUF in the MUFplex when no other MUFs are currently executing. The number is important. It should not be allowed to reach 4 billion, because at that number MUF terminates.

The line also gives the Log Block Sequence Number. This number starts at any number less than the number of tracks or blocks of the LXX when MUF is enabled. The same MUFplex rules as just previously given apply. The number is incremented for each log track or block with new log records. This number can only be reset by ending the MUF or all MUFplex MUFs and re-enabling it or them. The number is important. It should not be allowed to reach 4 billion, because at that number MUF terminates. This means that a busy MUF cannot remain enabled for years at a time.

Other Information

Other information that does not fit into one of the previously discussed categories is printed in another summary information report. For example, the following shows information that is printed if, in a z/OS environment, one or more of the high/low threshold options is used.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************
                                          OTHER INFORMATION


I/O THRESHOLDS - HIGH-260K LOW-132K MIN-1036K
```

Task Information

If the MUF fails with an abnormal EOJ, a page showing task information follows the previously shown other information report. The task information contains status information similar to other status displays, but the task information report is specifically targeted at helping to debug the abnormal failure. Fields unique to the task information report are described after the report example that follows.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************
                                          TASK INFORMATION

                                  ❶                                          ❷
JOBNAME  RUNIT TASK   TIME  I/O B-RF CMD-DBID-TBL STATUS          TSN    LOGR OWNR/Y VALUE/RID

JOB0040  79588    1    :04          UPDAT 997C02 PENDING
JOB0040  79588    2    :04        1 RDUNX 997F01 WAIT BFR LOCK D DBDB0771 83
JOB0040  79588    3    :04    1   2 RDUKG 997F02 WAIT BFR LOCK D DBDB076C101
JOB0040           4    :04          RDUNX 997F03 PENDING
```

❶

The B-RF field gives the current buffer references.

❷

The LOGR field gives the number of log records executed by the task since the last close or commit occurred.

Compound Boolean Selection (CBS) Information Example

The following Compound Boolean Selection (CBS) information section of the EOJ report contains information that compliments the already existing CBS statistics at EOJ. Following is an example of this CBS information section of the EOJ report:

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************
                                                CBS INFORMATION

                      ❶ SETS PROCESSED               561        SETS DELETED MAX AGE          0 ❷
                      ❸ SETS SPILLED                   0        SETS FETCHED                  0 ❹
                      ❺ CBSBFR CURRENT PERCENT FULL    0        CBSBFR MAX PERCENT FULL       0 ❻
```

❶

Sets Processed is the total number of sets processed by the MUF.

❷

Sets Deleted Max Age is the number of sets which were deleted from the CBS buffer due to the CBS MAX AGE setting.

❸

Sets Spilled is the number of sets which had to be paged to the CBS index. This is an interaction between the specified CBS buffer size and the number and size of open CBS sets. If Sets Spilled is non-zero, you might wish to either increase the CBS buffer size or examine your applications for missing SELPR statements.

❹

Sets Fetched is the number of times a spilled set was referenced to retrieve more rows. The ratio between Sets Spilled and Sets Fetched can be used to understand if sets that are truly in use are being spilled to the CBS index, or if it is an absence of SELPR in user programs which is causing sets, that are no longer in use, to be paged out.

❺

CBSBFR Current Percent Full shows the percent of CBS buffer max size that is being used by open sets.

❻

CBSBFR MAX Percent Full is the high water mark for concurrent open sets.

COMM NEWRXX (Request New Recovery File)

The COMM NEWRXX option requests a new Recovery File (RXX) be started. In response, the Multi-User Facility performs an IBM CLOSE/OPEN on the Recovery File, allowing operations to mount a **new** tape.

The COMM NEWRXX function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

When to Use

Use NEWRXX when using an active Recovery File (Multi-User startup option LOGRCV YES) and you want to close and demount the Recovery File (RXX) tape and begin recording information on a new tape.

Processing NEWRXX Requests

CA Datacom/DB ignores a NEWRXX request unless an active Recovery File (LOGRCV YES) is specified in the Multi-User startup options.

Execution of a NEWRXX request does not cause a spill to be performed. If a spill is in progress at the time of the request, Multi-User completes the spill before requesting a new tape. Examine the system log to determine the exact time of the last spill. Use the time of the last spill as an estimate in determining which Recovery File tapes are needed to perform a forward or backward recovery at some time in the future.

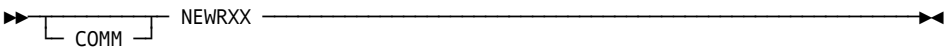
The Recovery File being demounted contains log records up to the point of the last spill executed by the Multi-User Facility. This does not necessarily mean that all tasks completed at the time of the spill were written on the Recovery File. For example, a long running job that is executing with transaction backout specified can prevent records from other completed tasks from being spilled.

CA Datacom/DB closes and reopens the Recovery File (RXX). Operations is responsible for mounting a new tape.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

COMM

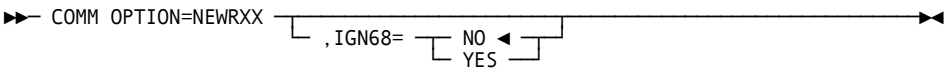
You can omit the COMM function name when submitting this console command.

NEWRXX

Requests the Multi-User Facility to begin recording Recovery File information on a new tape.

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keyword

OPTION=NEWRXX

Requests the Multi-User Facility to begin recording Recovery File information on a new tape.

Optional Keyword

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to issue the NEWRXX option.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
           COMM     OPTION=NEWRXX
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM NEWRXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          COMM OPTION=NEWRXX

                    FUNCTION=COMM
          ❷            OPTION=NEWRXX


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - COMM     NEWRXX   REQUEST ACCEPTED    ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue.

COMM REQABORT (Abort a Request)

The COMM REQABORT option provides a means of terminating various long running requests.

The COMM REQABORT function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

When to Use

Use this option to terminate a request that has been waiting or running a long time due to one of the following:

- Primary or secondary exclusive control
- Many read I/O events
- Waiting on a task

Requests which can impact wait time and for which this command is effective are LOCxx, REDxx, RDUxx, GETxx, CNTxx, and SELxx requests.

Also supported are ADDIT, DELET, and UPDAT commands. If one of these commands is subjected to REQABORT, the request, though aborted, is subject to a rollback to reverse any data area or Index Area changes made.

If commands are executing on behalf of an SQL request, the REQABORT might be accepted immediately or delayed. The decision to accept immediately or delay the REQABORT request is based upon the SQL process that is executing.

Processing a REQABORT Request

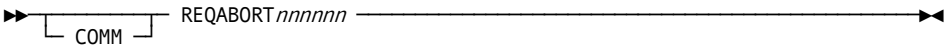
When you abort a request, CA Datacom/DB issues a return code 45 to the application program. Verify that programs contain instructions for handling the return code 45; for example, terminate further processing or have the program continue.

How to Use the Console Command

Issue a STATUS command to obtain the sequence number of the request causing the problem. For details, see COMM STATUS (Print Status of Multi-User Jobs) (see page 180). Use this request sequence number in the command to issue an abort request from the console.

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

COMM

You can omit the COMM function name when submitting this console command.

REQABORT

Requests an abort of a long running request.

Required Keywords

nnnnnn

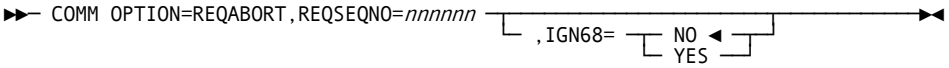
Sequence number of the request to be aborted.

How to Use the DBUTLTY Command

To execute the REQABORT function using DBUTLTY:

1. Perform a STATUS command to determine the sequence number of the request causing the problem.
2. Issue the following command to terminate the request.

Multi-User mode is required to execute this command.



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keywords

OPTION=REQABORT

Requests the Multi-User Facility to abort a particular request.

,REQSEQNO=

Indicates the internal request number for the request to be terminated.

Optional Keyword

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to abort a request number 124019.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          COMM     OPTION=REQABORT,REQSEQNO=124019
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM REQABORT

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          COMM OPTION=REQABORT,REQSEQNO=00282

                    FUNCTION=COMM
          ❷            OPTION=REQABORT
                       REQSEQNO=000282


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

DB00604I - COMM     REQABORT REQUEST ACCEPTED     ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue. If the request was not valid, an error message appears here.

COMM SNAP (Write Dump of Master List)

The COMM SNAP option writes a dump of the Master List to the Statistics and Diagnostics Area (PXX). The dump contains the Master List information. You can run this option at any time.

The COMM SNAP function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

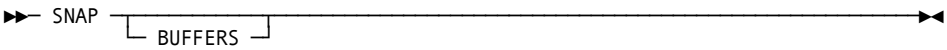
When to Use

Use this option at the direction of Technical Support. If you are having a problem with the Multi-User Facility, Technical Support personnel may ask for the information produced by SNAP to assist in problem determination. If the optional word BUFFERS is used, the Master List is produced with the standard CA Datacom/DB buffer area.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Note: Do **not** enter COMM with this console command.

Command

SNAP

Requests that the Master List be written to the Statistics and Diagnostics Area (PXX).

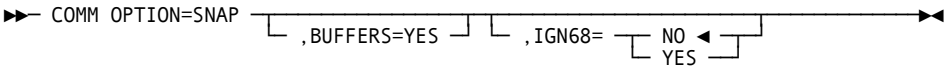
Optional Keywords

BUFFERS

Requests that the standard CA Datacom/DB buffer area be included.

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keyword

OPTION=SNAP

Requests that the Master List be written to the Statistics and Diagnostics Area (PXX).

Optional Keywords

,BUFFERS=YES

Requests that the standard CA Datacom/DB buffer area be included.

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to write a dump of the Master List with the standard CA Datacom/DB buffer area to the Statistics and Diagnostics Area (PXX).

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          COMM     OPTION=SNAP,BUFFERS=YES
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM SNAP

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          COMM OPTION=SNAP,BUFFERS=YES

                    FUNCTION=COMM
          ❷            OPTION=SNAP
                       BUFFERS=YES


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - COMM     SNAP REQUEST ACCEPTED       ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue. If the request was not valid, an error message appears here.

Master List Dump Example

The following example shows the first part of a Master List dump:

```
    MASTER LIST       REASON--ERROR 01(032)  JOB-69007-DBDVL     TASK-0001  3/20/2000  8.49.48    DUMP----------1
0865F000 D4C1E2E3 C5D940D3 C9E2E340 BA0C0EA0  0AD5051E 0000000A 0B627240 00001000  MASTER LIST .....N........ ....
0865F020 000000C8 0B6280C0 00001000 00000190  0B6310A0 00001000 00000000 FFFFFFFF  ...H............................
0865F060 04DB0008 B2E946A8 00000000 0865FC00  08661000 0B49A800 0B5BA548 30002400  .....Z..................$......
0865F080 00000000 0001B1EC 0001B254 0001B2BC  00000000 00000000 00000000 00000000  ................................
0865F0A0 00000000 00000000 00000000 00000000  00000000 0B489088 0B8B2000 0B5D3400  ................................
0865F0C0 00000000 00000000 04DB0003 00000000  8B419000 8B429000 8B426000 8B42D000  ..........................-.....
0865F0E0 00000000 00008000 8B44C000 8B43A000  8B441000 80016000 8B5AA000 8B445000  ......................-.......&.
0865F100 8B436000 8B449000 80024000 8B462000  8B421000 00000000 8B5AF000 8B466000  ..-.......  ..............0...-.
0865F120 8B46C000 00000000 8B431000 8B451000  00007EC0 8B432000 00000000 0B5B1008  ............................$..
0865F140 00006140 0B472020 00000000 00023E9D  0B5D1200 0B8B3000 00000000 00000000  ../ ............................
0865F160 0B5B4E80 8B400DD8 8B46B000 00000000  00000000 00000000 0B403598 0004B1B0  .$... .Q.................. ......
0865F1A0 0865FC20 0B5BA5A0 0004B000 0B402A70  08660020 0B416258 00000000 0B5BD020  .....$.......  ..............$..
0865F1C0 0B489E48 00000000 00000000 00000000  00060000 00660000 00660010 00100000  ................................
0865F1E0 00241000 00400000 00202000 25000001  00012A40 20000002 00000000 00000000  .....  .............  ............
0865F200 00000000 00000000 B2E91154 E6BB9280  00000000 00000000 B2E91103 B4DD4741  .........Z..W...........Z......
0865F220 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  ................................
0865F240 LINE(S) SAME AS ABOVE
0865F260 00000000 00000000 000267C0 0D400000  0865F26C 8B402111 00000000 00000000  ............. ....2.. ..........
0865F280 0B401FCE C4C2C4E5 D3404040 0002942F  00010001 C520F0F1 00200000 00000000  . ..DBDVL   ........E.01........
0865F2A0 00000000 00000000 00000000 00000000  00000000 DBDBBF6D 00000000 00000000  ................................
0865F2C0 00000000 00000000 000033AF C4C2C4E5  D4E4C640 00000000 00000000 00000000  ............DBDVMUF ............
0865F2E0 0000000A 00000000 0B41E524 96401324  96401000 0003F3FE 00000000 00000019  ..........V.. ... ....3.........
0865F300 00000000 00000000 00000000 00000000  00010000 00000000 00000010 00000000  ................................
0865F320 00000000 00000000 00000000 00190013  FFFC0000 0FF2942F 942F0000 00080063  ....................2...........
0865F340 0020001A 00080031 00300000 00020000  00000001 0000000F 001E0000 00000000  ................................
0865F360 00000004 00000000 00000000 00000000  04A40006 00000004 00001900 00006002  ............................-.
0865F380 0000F1F9 0E201900 004400C4 C2C4E5D4  E4C64011 01010000 041800C0 00000000  ..19.......DBDVMUF .............
0865F3A0 03E80014 00000000 00000000 00000000  00000000 00000000 0002942F 00029491  .Y..............................
0865F3C0 C3E7E700 03910004 8B457000 8B45A000  00000000 00000000 00000000 00000000  CXX.............................
0865F3E0 00000000 0B5D3080 00000000 00000000  00000000 00000000 00000000 00000000  ................................
0865F400 00000000 00000000 00000000 00000000  00000000 00000000 00000000 8B40B000  ............................ ..
0865F420 00000000 00000000 00000000 00000000  00000000 00000000 00000000 8B465000  ............................&.
0865F440 00000000 00000000 0B906000 00000000  808C1BC8 00000000 40000000 00000000  .........-........H.... .......
0865F460 00000000 00000000 00000000 8B59B000  8B5A0000 00000000 00000000 99092708  ................................
0865F480 48220000 00009909 27084853 00000000  8B5A3000 00000000 00000000 0004E2B6  ............................S.
0865F4A0 DBDBDBDB 10D500A0 0865F6D0 00000000  00000000 00000000 00800001 000264A0  .....N....6.....................
0865F4C0 00000000 F9F9F9F0 00001324 00000000  00000000 00000000 00000000 00000000  ....9990........................
0865F4E0 00000000 00000000 00000000 00000000  00000000 00000000 0B48D000 00000001  ................................
0865F500 8B417000 0B640AC0 0B640B60 0B640C00  0B641880 0B641920 00000000 F2F0F0F0  ..........-.............2000
0865F520 F0F50063 00000000 00000000 00000000  00000000 00000000 00000000 00000000  05..............................
0865F540 00000000 00000000 00000000 00000000  00000000 00000000 00000000 00000000  ................................
0865F580 LINE(S) SAME AS ABOVE
              .
              .
              .
```

COMM STATS (Write Current Statistics)

Using COMM OPTION=STATS, a database that is open can have the current statistical information written to the Directory (CXX), index, and data control blocks.

When to Use

A STATS function is for open databases that are not VIRTUAL or SYSTEMDBID. Review message DB00622W for databases that could not be processed. For details about message DB00622W, see the *CA Datacom/DB Message Reference Guide*.

A STATS function done through the console or DBUTLTY is accepted and processed without messages unless it is a virtual database, in which case a **DB00611I - COMM xxxxx AGAINST VIRTUAL yyyyyyyy IGNORED** message is issued. A request for a specific database that is virtual is ignored. If all databases were requested, any of them that are virtual databases are ignored.

Note: A virtual database is a *memory only* facility. It cannot be closed nor can its statistics be written to DASD.

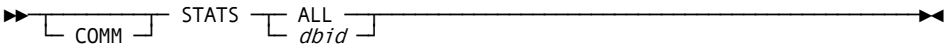
COMM STATS Flushes Pipeline

The STATS option of the COMM function flushes the pipeline for the database as part of pushing the STATS to the CXX, index, and data sets.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



COMM

You can omit the COMM function name when submitting this console command.

STATS

Requests that current statistics be written.

Required Keywords

ALL

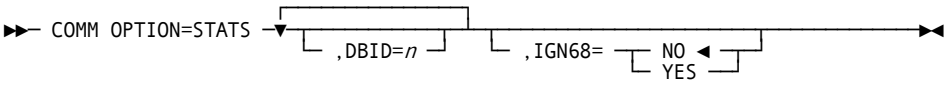
Specify ALL, if you want all databases to be processed. If you request that STATS be written for all databases, STATS are written for all databases that are *not virtual* databases. If any databases are virtual databases, a warning message is issued.

dbid

Specify a database ID, if you want only one particular database to be processed. If you request STATS for a particular database and the database is virtual, a warning message is issued, and no stats are written.

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute the COMM function with the following command format.



COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keyword

OPTION=STATS

Requests current statistics be written.

Optional Keywords

,DBID=*n*

DBID= is optional and may repeat. Each occurrence may specify one or more valid database IDs or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250).

If DBID= is not specified, all databases are processed.

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database in a range not defined is ignored.

If you request STATS for a particular database and the database is virtual, a warning message is issued and no stats are written. If you request (by not specifying a DBID) that STATS be written for all databases, STATS are written for all non-virtual databases. If any databases are virtual, a warning message is issued.

Valid Entries:

One or more valid database IDs or a range of DBIDs as already described.

Default Value:

ALL

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to write current statistics.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          COMM      OPTION=STATS
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM STATS

Here is a sample COMM OPTION=STATS report.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
           ❶         COMM OPTION=STATS

                    FUNCTION=COMM
           ❷           OPTION=STATS


           ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - COMM     STATS    REQUEST ACCEPTED    ❹
```

❹

Message indicating that DBUTLTY successfully placed the request on the Multi-User communication queue.

COMM STATUS (Print Status of Multi-User Jobs)

The COMM STATUS option displays the status of all jobs operating through the Multi-User Facility. If a job is waiting, the status report shows the reason. When executed through DBUTLTY, that is, COMM OPTION=STATUS, the utility job is always listed in the report with a status of ACTIVE, except in z/OS with a DBSIDPR not specifying XCF.

The COMM STATUS function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

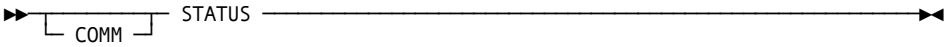
When to Use

Use the COMM STATUS option when you want to display the status of a job.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

COMM

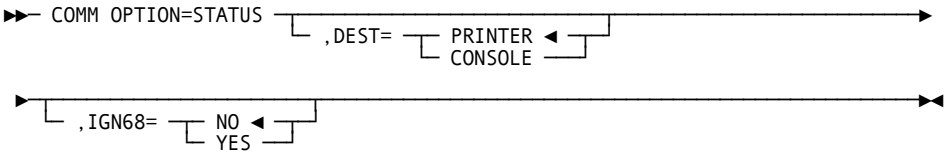
You can omit the COMM function name when submitting this console command.

STATUS

Requests a display of the current tasks in the Multi-User Facility.

How to Use the DBUTLTY Command

You can execute this command in either Single User or Multi-User mode. Execute the COMM function with the following command format:



COMM

Invokes the function to pass communication to the Multi-User Facility.

Required Keyword

OPTION=STATUS

Requests that the current status of tasks in Multi-User Facility be displayed. COMM OPTION=STATUS in Version 12.0 (see COMM STATUS (Print Status of Multi-User Jobs) (see page 180)) uses the SID where a SID is understood to be a sub-identifier ID, to determine the specific Multi-User Facility to access.

Optional Keywords

,DEST=

Specifies the destination for the status report. CONSOLE causes the report to appear on both the console and SYSLST. PRINTER causes the report to appear on SYSLST only.

Note: When routing print to the console, headings do not appear.

Valid Entries:

CONSOLE (may be abbreviated as CON) or PRINTER (may be abbreviated as PRI)

Default Value:

PRINTER

IGN68=

Specify IGN68=YES to ignore a return code 68. IGN68= is allowed for every ACCESS and COMM function. By ignoring return code 68, you can set up and run JCL that incorporates a stack of DBUTLTY functions without requiring that the MUF be enabled. IGN68= would therefore be useful, for example, if you wanted to load a database without first enabling the MUF, perhaps to allow restart to properly occur. Following is an example showing a stack of DBUTLTY functions that use IGN68=.

```
ACCESS STATUS=UTLTY,DBID=997,USERS=WAIT,IGN68=YES
COMM   OPTION=CLOSE,DBID=997,IGN68=YES
INIT   AREA=IXX,DBID=997
INIT   AREA=A01,DBID=997
LOAD   AREA=A01,DBID=997,FORMAT=NONE,KBYTES=9999,SORT=1
ACCESS STATUS=WRITE,DBID=997,IGN68=YES
```

Note: If the CXX is externally secured, the INIT and LOAD functions do not execute without the MUF being available, and IGN68=YES is therefore not recommended.

A function or functions specified with IGN68=YES generates message DB13001E to note that the function did not occur. Because no error was recognized, however, other functions in the stack continue to execute, and the DBUTLTY step ends without a recognizable error. Message DB13001E signals that the MUF was not enabled For details about DB13001E, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to print the status of all jobs operating through the Multi-User Facility.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
           COMM     OPTION=STATUS
 /*
 /&
 * $$ EOJ
```

Sample Report: COMM STATUS Utility

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          COMM OPTION=STATUS

                    FUNCTION=COMM
          ❷              OPTION=STATUS


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

DB13101I - STATUS  MULTI-USER AVAILABLE     , SUBID=3           ❶

     ❷     ❸    ❹    ❺             ❻      ❼   ❽      ❾   ❿
 JOBNAME R-UNIT TASK STATUS          CMD    DBID TBL    SEQ TIME       I/O ID                TSN        OWNR/Y TSN-TM

 DBDVL132 14160    1 READY TO RUN     ADDIT  997 C02  95727                                  DBDBBBDA             :01

 JOBNAME  R-UNIT TASK VALUE/RID
 STATJOB2  33654    1     1 PAY
```

❶

Messages indicating the Multi-User job status.

❷

JOBNAME is the name of the active job currently connected to the Multi-User Facility or, if a system task, one of the following:

*****ACTNG**

Accounting

*****DBCOM**

Tasks that process a console request, such as COMM CLRPXX

*****DBOPN**

MUF immediately after the enable to allow accounting, and other system tables the opportunity to open

*****DBRST**

RESTART function during the enable of the Multi-User Facility

*****DBSYS**

Rare systems functions

*****HISTR**

History database spill

*****IOCMP**

Asynchronous I/O completion system processing

*****QUEUE**

Index queue

*****SCRTY**

Security

*****SPILL**

Log spilling

❸

R-UNIT is the run unit number, where nnnnn is the low order five digits with leading zeros suppressed (nnnnn is zero for system tasks).

❹

TASK is the task number nnnn (RWTSA ID) where the task is running.

❺

The status messages occur in groups based on task status. STATUS can be one of the following:

ABENDING

The task has detected a task abend.

ACCESS AREA SMP

An ACCESS with an AREA specified is running and interacting with processes that care about the ACCESS AREA function.

ACTIVE

A request is active (only one task can be active).

BUFFER, CXX

A task is waiting on a specific Directory (CXX) block to become available.

BUFFER, DATA

A task is waiting on a specific data area block to become available.

BUFFER, DXX

A task is waiting on a specific Index Area (DXX) block to become available.

BUFFER, IXX

A task is waiting on a specific Index Area (IXX) block to become available.

BUFFER, LXX

A task is waiting on a specific Log Area (LXX) block to become available.

CICS COLD START

This task has an outstanding CICS two-phase commit, and CICS has been cold started. The task must be manually researched and then cleaned up with a REQCOMIT or REQROLBK.

CICS IN DOUBT

A task is waiting on CICS to provide resolution for a two-phase commit.

CLOSE, aaabbbbb

The task is waiting on the close of the named area-database (aaabbbb).

DEQ/UNLOCK CXX

Reflects that the task is waiting on the MVS DEQ (for MVS) or z/VSE UNLOCK (for z/VSE) of the CXX.

ENQ/LOCK CXX

Reflects that the task is waiting on the MVS ENQ (for MVS) or z/VSE LOCK (for z/VSE) of the CXX.

EOJ IN PROCESS

An EOJ is in process.

LOG CHAIN LOCK

A task is waiting on the log chain.

LOG FORCE BLOCK

A task is waiting for the read forced list lock in a MUFplex environment.

NOT ACTIVE

The status is not active.

OPEN, aaabbbbb

The task is waiting on the open of the named area-database (aaabbbbb).

PENDING

A request has been transmitted but has not yet been accepted by the Multi-User Facility.

POSTED

The Multi-User Facility has accepted the task but has not dispatched it because of priority.

PRC TCB RUNNING

A procedure is running on behalf of this task in a procedure TCB.

READY TO RUN

The resource upon which the task was waiting has become available. Task is ready to be dispatched, based on priority.

SYS STATS FLUSH

A task is waiting on a close lock that controls flushing statistics to the PXX.

TASK ACTIVE

The CICS transaction has a locked thread but is not currently waiting on a Multi-User Facility request.

WAIT ACCT ENTRY

A task is waiting to update a particular entry in the Accounting buffer.

WAIT ACCT GLOBL

A task is waiting on the Accounting global chain.

WAIT ACCT TABLE

A task is waiting on a specific Accounting table to record or remove information.

WAIT ACTG

A task is waiting on the Accounting Facility.

WAIT ADD LOG RC

A task is waiting on access to current log buffer.

WAIT BUFFER

A task is waiting for a free buffer (CXX, LXX, IXX, DXX, data).

WAIT CBS

A task is waiting on access to the CBS buffer lock.

WAIT CBSBFR SPL

A task is waiting for a set in the CBS buffer to spill to the IXX.

WAIT CICS START

A task is waiting on CICS to provide resolution for a two-phase commit, but the controlling CICS is not up.

WAIT CMPUS EXIT

A task is waiting on a non-RENT user compression exit.

WAIT DATA BF SP

A task is waiting on the data free space buffer chain lock.

WAIT DATA SP MG

A task is waiting on the Data Space Management lock.

WAIT DEFRAG

DEFRAG is running and interacting with a read backward request.

WAIT E/C

A task is waiting for a record that another task has read with exclusive control. The status is followed by OWNER- indicating the task owning the resource and by RID- indicating the type of lock. See the description of VALUE/RID on page Sample Report: COMM STATUS Utility.

WAIT HASH DATA

A task is waiting on the hash chain for a data area buffer.

WAIT HASH DXX

A task is waiting on the hash chain for a DXX buffer.

WAIT HASH IXX

A task is waiting on the hash chain for an Index Area buffer.

WAIT I/O

A task is waiting on an I/O event.

WAIT INDEX QUE

A task is waiting on access to the index queue.

WAIT INTERLCK D

A task is waiting on the interlock detection lock.

WAIT LOCK LIST

A task is waiting on access to the exclusive (share) list control tables.

WAIT LOG DELAY

A task is waiting on log delay based on your specifying X_LOG_DELAY_BLK.

WAIT LRU CXX

A task is waiting on the least recently used chain for a Directory (CXX) buffer.

WAIT LRU DATA

A task is waiting on the least recently used chain for a data area buffer.

WAIT LRU DXX

A task is waiting on the least recently used chain for a DXX buffer.

WAIT LRU IXX

A task is waiting on the least recently used chain for an Index Area buffer.

WAIT LRU LXXTXB

A task is waiting on the least recently used chain for a Log Area (LXX) buffer for transaction backout.

WAIT MEM-MGR

A task is waiting on access to the Memory Manager.

WAIT NEWRXX

A COMM NEWRXX function is waiting on the open/close subtask to open or close the RXX.

WAIT OFT

A task is waiting on access to the open list of tables lock.

WAIT OPEN EXIT

A task is waiting on open exit.

WAIT POST LIST

A task is waiting on access to the posted list.

WAIT PRC CHAIN

A task is waiting on the procedure chain processing lock.

WAIT PROC TCB

A task is waiting for a PROCEDURE TCB. All procedure TCBs are currently in use.

WAIT PXX LOCK

A task is waiting on access to the Statistics and Diagnostics Area (PXX) write module.

WAIT REQ

A task is waiting on a request in another task area to complete before executing a spill.

WAIT SECURITY

A task is waiting on security validation.

WAIT SPILL

A task is waiting on a spill to complete.

WAIT SQL ATTACH

A task is waiting on access to the SQL ATTACH list lock.

WAIT SQL GLOBAL

A task is waiting on access to the SQL GLOBAL list lock.

WAIT SQL MEMORY

A task is waiting on an SMP lock around the SQL call to memory manager.

WAIT SQL MISC

A task is waiting on the SQL miscellaneous lock.

WAIT SQL PLAN

A task is waiting on access to the SQL PLAN list lock.

WAIT SQL STATUS

A task is waiting on access to the SQL STATUS list lock.

WAIT SQL TTM

A task is waiting on access to the TTM table lock.

WAIT SQL URT

A task is waiting on access to the SQL URT list lock.

WAIT TASK AREA

A task is waiting on the task area lock for status change.

WAIT TASK

A task is waiting on record owned by another task. The status is followed by OWNER- indicating the task owning the resource and by RID- indicating the type of lock. See the description of VALUE/RID on page Sample Report: COMM STATUS Utility).

WAIT TASK DELAY

QUIESCE is waiting on an active request.

WAIT TIMER LOCK

A task is waiting on the timer and cancel lock.

WAIT TSN

A task is waiting on the TSN lock to check, assign, or remove an active transaction.

WAIT USER EXIT

A task is waiting on another user exit to complete.

WAIT W/P CH DAT

A task is waiting on the write pending chain for a data area buffer.

WAIT W/P CH IDX

A task is waiting on the write pending chain for an Index Area buffer.

WAIT WAIT LIST

A task is waiting on access to the wait list.

❻

CMD indicates the command in the Request Area. This is blank for a system task or omitted if the request is NOT ACTIVE, TASK ACTIVE, PENDING, or POSTED.

❼

DBID is a database ID (not listed when not applicable).

❽

TBL is a table name (not listed when not applicable).

❾

SEQ specifies the internal CA Datacom/DB request sequence number. This allows you to abort a request if it is blocking other requests from executing. For a system task, zero is displayed. It is omitted if the request is NOT ACTIVE, TASK ACTIVE, PENDING, or POSTED.

❿

TIME indicates the length of time in minutes and seconds since the Multi-User Facility accepted the request. For system tasks, this is the time since the system started the current work unit. It is omitted if the request is NOT ACTIVE or TASK ACTIVE.

(Page 2 continued)

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                                General Utility                               *      Release: 12
                    *                    Copyright © 2009 CA. All rights reserved.                 *               SPnn
                    ********************************************************************************

DB13101I - STATUS  MULTI-USER AVAILABLE     , SUBID=3

                                                                          ❶  ❷                ❸             ❹
 JOBNAME R-UNIT TASK STATUS          CMD    DBID TBL    SEQ TIME       I/O ID                TSN        OWNR/Y TSN-TM

 DBDVL132 14160    1 READY TO RUN     ADDIT  997 C02  95727                                 DBDBBBDA             :01
   ❺         ❻     ❼         ❽
 JOBNAME  R-UNIT TASK VALUE/RID
 STATJOB2  33654    1     1 PAY
```

❶

I/O indicates the number of I/O events performed during the processing of this request. It is omitted if the request is NOT ACTIVE, TASK ACTIVE, PENDING, or POSTED.

❷

ID is displayed (xxxxxxxx or tran-term-opr-xxxxx) under the following circumstances:

- Under CA Datacom Server 5.0 or later, the ID field contains an 8-character transaction identifier and the 8-character name associated with the PC.
- If a CICS transaction has a request active or if CICS has a transaction without an active request but a locked task area, the following are displayed:

  tran The CICS transaction identifier

  term The CICS terminal ID

  opr The CICS operator ID

  xxxxx CICS sequence number
- If executing a DBUTLTY function with a request in the Multi-User Facility, the field provides the literal 'DBUTLTY ' followed by the function being executed (except DBTEST PGM=DBTSTPR, which is treated as an easy test program and not as a privileged utility function).

❸

TSN (only listed on updates).

❹

OWNR/Y TSN-TM is the task of the owner (OWNR) and the number (Y) of the Multi-User Facility in the MUFplex upon which the task is waiting. TSN-TM refers to time since the transaction was started. TSN-TM is provided by DBUTLTY STATUS *not* console.

❺

When you issue a COMM STATUS command, CA Datacom/DB displays status information for tasks that are waiting, where JOBNAME is the name of the active job currently connected to the Multi-User Facility or, if a system task, to one of the entries listed in the table for the JOBNAME column on Page 2 of the report.

❻

R-UNIT is the run unit number, where nnnnn is the low order 5 digits with leading zeros suppressed (nnnnn is zero for system tasks).

❼

TASK is the task number nnnn (RWTSA ID) where the task is running.

❽

VALUE/RID indicates a value lock. The following table identifies the types of locks that can occur. If a task status is WAIT E/C or WAIT TASK, different types of locks can be responsible for the wait. When the lock is a record lock or a table lock, DBUTLTY prints a record ID in the text of a report message line numbered DB01329I to indicate the resource causing the lock that is responsible for the wait, as described in the following chart:

Lock Type	RID Format	Meaning
Record	tid-rrrrrrrrrrrrr	In the RID format, *tid* is the Table ID and *rrrrrrrrrrrrr* is the Record ID.
Internal	iiiiFFjjjjjjjjjjj	Provides internal information.
Table	ddddtttt	The *dddd* is the Database ID and *tttt* is the Table ID. The remaining five bytes contain zeros.

When the lock is a value lock, CA Datacom/DB provides additional information as shown in the following table:

Type of Value Lock	Meaning	Additional Information Available from the COMM STATUS Display
DBSCXX	The Directory (CXX) is locked during URT open and close processing and during maintenance to the CXX when CXX I/O is required.	(None)
DBSDCB	CA Datacom/DB has locked this area during a dynamic extend.	Database ID and Area name
DBSDDL	SQL Data Definition Language (DDL) has locked the process of choosing internal IDs for various structures.	Database ID where the created or altered table resides.
DBSDSB	CA Datacom/DB has locked a database to ACCESS OFF/READ which is open.	Database ID
DBSIXX	CA Datacom/DB has a lock while index que/backward read processing.	Database ID and Block group number

Type of Value Lock	Meaning	Additional Information Available from the COMM STATUS Display
DBSKEL	A table's key element data is locked during the process to bring it into memory or write it back to the Directory (CXX).	Database ID and Table name
DBSKEY	CA Datacom/DB has locked a key to verify that an add or update does not violate a unique constraint.	■ Database ID ■ Table ID ■ Key ID ■ Key value to be added/updated.
DBSMVR	A user has executed the DBUTLTY LOCK function to block moved records.	Database ID and Area name
DBSOPN	CA Datacom/DB has a lock during URT open and close processing and during maintenance to the CXX.	Database ID
DBSPLN	CA Datacom/DB has locked an SQL plan for concurrency.	AUTHID and Plan name
DBSPRC	The locking of a prodecure has occurred so that a NEWCOPY can be synchronized.	Eight-character procedure name
DBSRRD	CA Datacom/DB has locked a key range to assure the SQL repeatable read function.	■ Database ID ■ Table ID ■ Key ID ■ Key range
DBSSIX	CA Datacom/DB has locked the data space management index during maintenance.	■ Database ID ■ Area name ■ Block group number
DBSSOC	CA Datacom/DB has locked a database and table to open or confirm the table for SQL processing.	Database ID and Table name

Type of Value Lock	Meaning	Additional Information Available from the COMM STATUS Display
DBSSPL	CA Datacom/DB has a lock during the spill.	(None)
DBSTXB	CA Datacom/DB has a lock on this transaction while waiting on the rollback to be completed using backward recovery. A previous console message **DB00109W** requested the recovery.	Transaction sequence number
DBSURI	CA Datacom/DB has locked an area during URI validation/assignment.	Database ID and Area name
Other	Other lock values are possible.	

In Version 12.0 (but not in the release previous to it), the following is also printed per task with the JOBNAME, R-UNIT, and TASK fields (as in the previous release), followed by the (new for Version 12.0) status information fields, BREF, R-TIME, C-TIME, D-TIME, and USER-ID as shown in the following example and explained in the following descriptions:

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                             Multi-User Facility                              *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: DBDVMUF

                         ❶     ❷      ❸      ❹     ❺     ❻      ❼         ❽      ❾
 JOBNAME R-UNIT TASK BREF R-TIME C-TIME D-TIME W-TIME  PATH U-SYSTEM  U-JOBID USER-ID

 DBDVLSTS 44431    1                  *NULL* *NULL*  UNKN CA11     JOB39119 COMM
```

❶

BREF is the number of buffer references to satisfy the current request (SQL, set-at-a-time, record-at-a-time). This is the total for index and data. It is printed in the same format as the I/O column.

❷

R-TIME is the run time in minutes and seconds that the current request (SQL, set-at-a-time, record-at-a-time) has been set by MUF to be active.

❸

C-TIME is the CPU time the current request (SQL, SAAT, RAAT) had completed the last time it allowed a task switch. It is only available if accounting is enabled in the MUF, and if accounting CPU time is also currently set. The format of the field is minutes and seconds (*mmm*:*ss*) with two exceptions. A time greater than 999 minutes is reported as +++:++. A time less than one second is reported as blanks, not as 000:00.

❹

D-TIME is the duration time the task has been waiting for that upon which, for this instance, it is currently waiting. It is not the total wait time for the current request, nor is it the wait time for the current data manager request. If the task is not waiting, the field is set to null, and on the report it is printed as the characters *null*. The format of the field is minutes and seconds (*mmm*:*ss*) with two exceptions. A time greater than 999 minutes is reported as +++:++. A time less than one second is reported as blanks, not as 000:00.

❺

W-TIME is the total wait time that this request was not selected by the MUF for processing, excluding a current wait, if any, whose time is in D-TIME.

❻

PATH is the path from the user application to the MUF and can be as follows:

LOCAL

LOCAL is the PATH if the user and the MUF are on the same system. If there is no user, that is, the request was generated by the MUF itself, for example from being restarted or by the execution of a console or console-like command.

XCF

XCF is the PATH if the z/OS XCF facility is being used for communication.

CCI

CCI is the PATH if the CA CCI facility is being used for communication.

IUCV

IUCV is the PATH if the z/VSE IUCV facility is being used for communication.

UNKN

UNKN is the PATH if the path cannot be determined (this PATH is not expected to occur).

❼

U-SYSTEM is the system name, where the user application called the interface.

❽

U-JOBID is the z/OS JOBID identification of the user application.

❾

USER-ID is the 32-character user ID, if this request was made through the DBNTRY entry point.

Sample COMM STATUS Online

```
    .
    .
    .
DB01311I - STATUS
DB01327I - Multi-User AVAILABLE, Tasks - Attached----63, Available----87
DB01328I - TIME   I/O  JOBNAME  R-UNIT TASK CMD-DBID-TBL    SEQ STATUS         ID                      OWNR/Y TSN-TM
DB01329I -    :06  257 ***SPILL         157 SPILL               WAIT I/O
DB01329I -             QA#@T796  43571   71                    TASK ACTIVE     MLT:-T004-DB:-35651
DB01329I -             QA#@T796  43571   73                    TASK ACTIVE     MLT:-T006-DB:-67725
DB01330I -             QA#@T796  43571   68 NOT ACTIVE(0002)
DB01329I -             QA#@T796  43571   70 DELET 796F01 678444 READY TO RUN    MLT:-T003-DB:-48157
DB01329I -           1 QA#@T796  43571   74 ADDIT 796K01 678392 WAIT I/O        MLT:-T007-DB:-16740
DB01329I -           2 QA#@T796  43571   72 ADDIT 796F03 678393 WAIT I/O        MLT:-T005-DB:-21126
DB01329I -             QA#@T797  43572   77                    TASK ACTIVE     MLT:-T003-DB:-28504
    .
    .
    .
DB01329I -             T784TST1  43561   23 ADDIT 784F03 736153 WAIT E/C DBSMVR MLT:-T005-DB:-18420      21
DB01329I -             T784TST1  43561   24 ADDIT 784RNA 736160 WAIT E/C DBSMVR MLT:-T006-DB:-25068      21
DB01329I -             T784TST1  43561   21 ADDIT 784F01 736183 WAIT E/C DBSMVR MLT:-T003-DB:-22024      21
DB01329I -             T787TST1  43566   42                    TASK ACTIVE     MLT:-T006-DB:-29239
DB01329I -             T787TST1  43561   39 ADDIT 784F01 736564 READY TO RUN    MLT:-T003-DB:-27574
    .
    .
    .
JOB05796  DBDVMUF7:DB01331I - JOBNAME  R-UNIT TASK VALUE/RID
JOB05796  DBDVMUF7:DB01332I - T784TS1   43561   23   784 F03
JOB05796  DBDVMUF7:DB01332I - T784TS1   43561   24   784 RNA
JOB05796  DBDVMUF7:DB01332I - T784TS1   43561   21   784 F01
```

Chapter 7: CONFIRM (Verify Constraint Integrity)

Constraint integrity can be verified by using the CONFIRM function of the CA Datacom/DB Utility (DBUTLTY).

CONFIRM

The Multi-User Facility must be initiated to perform this function.

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When to Use

Use the CONFIRM function when one or both of the following conditions indicate that the integrity of the data in a table must be verified:

- An application receives a return code indicating the table has a CHECK PENDING status.
- A Directory (CXX) report shows the tables to have a CHECK PENDING status. See Directory (CXX) Reports for an example.

The DBUTLTY functions which can result in a CHECK PENDING status for a table are the LOAD function (for details, see LOAD (Build a Data Area) (see page 393)), the LOAD portion of the REORG function (for details, see REORG (Parallel BACKUP) (see page 475)), and the RECOVERY function (for details, see RECOVERY (Rebuild a Database) (see page 443)).

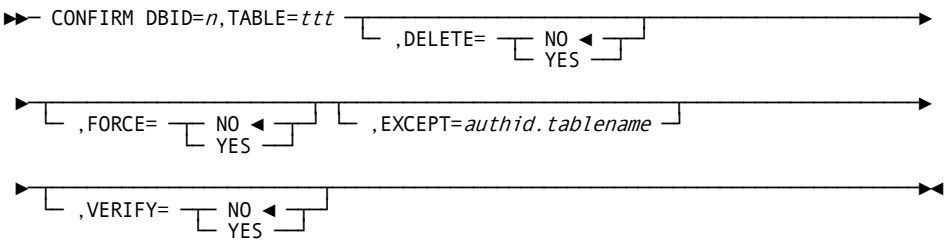
Note: You must CONFIRM tables in the following order, from referenced-to-referencing, or parent-to-child, that is to say, the referenced table cannot be in check pending status. For example, if there is a foreign key from a REMARKS table to a LINE_ITEM table on PO_Nbr and Line_Item_Nbr, and the LINE_ITEM table has a foreign key that references the PO table on PO_Nbr, the PO table must be loaded first, then the LINE_ITEM table can be confirmed, then the REMARKS table can be confirmed. Therefore, the confirm order would be: from PO table to LINE_ITEM table to REMARKS table.

Determining Constraints In Effect

To determine the constraints in effect for a table, query the SYSCONSTRSRC table using SQL statements. (See the *CA Datacom/DB Database and System Administration Guide* for details on querying the SYSCONSTRSRC table.) .

How to Use

Multi-User mode is required to execute this command. To confirm a table's constraints, execute DBUTLTY with the command:



Command

CONFIRM

Invokes the function to verify the integrity constraints for a table.

Required Keywords

DBID=

Identifies the database containing the table.

Valid Entries:

DATACOM-ID of the database containing the table

Default Value:

(No default)

,TABLE=

Identifies the table to be verified.

Valid Entries:

DATACOM-NAME of the table

Default Value:

(No default)

Optional Keywords

,DELETE=

Specifies whether invalid records (rows) in the table are to be deleted.

Valid Entries:

NO or YES

Default Value:

NO

,EXCEPT=

Specifies an SQL table into which the rows found to violate constraints are inserted. If DELETE=YES is also specified, the records (rows) are saved into this table before they are deleted from the table being confirmed.

Valid Entries:

The name of an existing SQL table whose definition matches that of the table being confirmed. You must specify an AUTHID.

Default Value:

(No default)

,FORCE=

Allows you to bypass constraint verification and move the table from CHECK PENDING to usable status regardless of the underlying data. Use of this option is recorded in the FORCED OFF field of the HISTORY CHECK PENDING section of the CXX Report. Use of this option is valid only when the user is certain that the data contained in the table conforms to all constraints.

Valid Entries:

NO or YES

Default Value:

NO

,VERIFY=

Indicates that you want the data to be checked, even if the table is not in CHECK PENDING status.

For example, you may have executed the CONFIRM function, specifying FORCE=YES which does not check the data, but indicates that you have forced the CONFIRM. When you use the VERIFY option, the system confirms the data, checking for integrity constraints.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

This example confirms table DAT in database 30 and records exceptions to table DB.OPEN.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
   CONFIRM DBID=30,TABLE=DAT,VERIFY=YES,DELETE=YES,EXCEPT=DB.OPEN
 /*
 /&
 * $$ EOJ
```

Sample Report: CONFIRM

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          CONFIRM DBID=30,TABLE=DAT,VERIFY=YES,DELETE=YES,EXCEPT=DB.OPEN

                    FUNCTION=CONFIRM
                       DBID=30
          ❷            DELETE=YES
                       EXCEPT=DB.OPEN
                       TABLE=DAT
                       VERIFY=YES


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing. This area can also return an error code related to processing the CONFIRM request. If an error occurs during CONFIRM processing, interpret the error code and the internal return code of CA Datacom/DB error code 94 (see the *CA Datacom/DB Message Reference Guide*).

Chapter 8: CXXMAINT (Maintain Directory)

The CXXMAINT function of DBUTLTY maintains and updates your CA Datacom/DB Directory (CXX).

Overview

Use the CXXMAINT function when you want to:

- Activate/deactivate a key for Compound Boolean Selection use
- Alter the DSN specified in the Directory
- Activate/deactivate logging
- Convert the Directory to the current version
- Update Directory definitions from CA Datacom/DB
- Delete definitions that are no longer useful
- Alter the space reclamation option
- Identify that a specified database does not exist to minimize recovery

Each option is discussed separately in the following sections of this chapter.

Important! Do **not** use the CXXMAINT DDPROD option for SQL accessible tables, or on a database containing SQL accessible tables, or on databases containing partitioned tables. Use the CA Datacom/DB CATALOG transaction instead.

The CXXMAINT function protects you from destructive Directory (CXX) modification. If you run the function while any task is active against the target database(s), CXXMAINT terminates without performing any modification with the following exceptions. You can use the following CXXMAINT ALTER options with a database open in a Multi-User Facility:

- CBSUSE
- DSN
- DSOP
- LOGGING

You can also use CXXMAINT DELETE with a database open in a Multi-User Facility as long as the specified table has no open User Requirements Tables. The ACCESS function can still be used, if desired, to prevent tasks from opening a database while you are running a CXXMAINT function. For more information, see ACCESS (Restricting Opens or Maintenance) (see page 55).

Frequent backups of the Directory are recommended. When a hardware failure or accidental destruction occurs, a current backup provides a quick method of restoring the Directory. If the backup is not current, much time can be lost in determining what maintenance transactions must be applied to bring the Directory up to date. See BACKUP CXX (Back Up Directory) (see page 89) for details.

CXXMAINT ALTER CBSUSE (Trial Delete a Key from CBS Use)

The ALTER CBSUSE option allows you to specify whether a particular key is used by the Compound Boolean Selection Facility for SELxx commands. If a key is deleted from Compound Boolean Selection use, it remains available for record-at-a-time requests and CA Datacom/DB maintains it during ADDIT, UPDAT, and DELET processing. Cataloging the database restores the use of all keys.

When to Use

Use the ALTER CBSUSE option to test how deleting a key would change performance of applications accessing the specified table. Performance can improve as well as degrade.

Note: This function is not valid for use with the Native Key of a table.

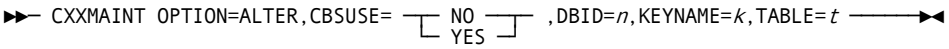
You can use this option with a database open in a Multi-User Facility.

Effects on Maintenance

Adds or deletes of Index entries during maintenance continue during the trial deletion of a key.

How to Use

Multi-User mode is required to execute this command. To mark a key for trial delete, use the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

CBSUSE=

Indicates whether the key can be used for Compound Boolean Selection.

Valid Entries:

NO or YES

Default Value:

(No default)

,DBID=

Identifies the database containing the subject key.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,KEYNAME=

Specifies the key which is to be flagged for trial deletion.

Valid Entries:

DATACOM-NAME of the subject key

Default Value:

(No default)

,TABLE=

Identifies the table in which the key is located.

Valid Entries:

DATACOM-NAME of the table

Default Value:

(No default)

Example JCL

The following shows the command to disable a key named POLI in table POL from use by the Compound Boolean Selection Facility.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
         CXXMAINT   OPTION=ALTER,CBSUSE=NO,DBID=1,KEYNAME=POLI,TABLE=POL
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT ALTER CBSUSE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
         ❶           CXXMAINT OPTION=ALTER,CBSUSE=NO,DBID=1,KEYNAME=POLI,TABLE=POL

                    FUNCTION=CXXMAINT
                       CBSUSE=NO
         ❷             DBID=1
                       KEYNAME=POLI
                       OPTION=ALTER
                       TABLE=POL

         ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2


                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************* Directory: PRODCXX

          ❹         *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT ALTER DBCS (Activating DBCS Support)

The ALTER DBCS option allows you to specify what kind of double-byte character set support you would like for your Directory (CXX). Also see CXXMAINT ALTER LANGUAGE (Activating LANGUAGE Support) (see page 218).

Note: If you set your environment for DBCS or Katakana data, entity-occurrence names and data that contains DBCS or Katakana data may be displayed on a terminal in CA Datacom/DB and CA Dataquery. However, if a terminal does not support the display of DBCS or Katakana data, the terminal may display the data incorrectly or a terminal error such as a PROG402 abend can occur.

When to Use

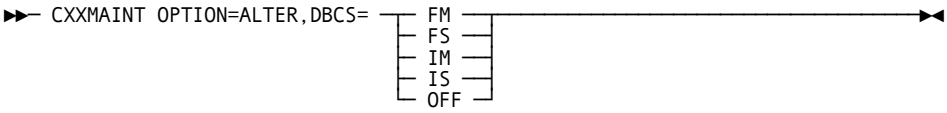
Use the ALTER DBCS option to enable and specify the type of DBCS support or to disable the DBCS support.

How to Use

Multi-User mode is required to execute this command.

Note: If you are using SQL, you must restart the MUF after running this command.

To activate DBCS support, use the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,DBCS=

Indicates whether to use a double-byte character set and what type.

FM

Indicates that Fujitsu standard shift codes (28,29) are to be used for mixed data and that new character fields are to be defaulted to have the MIXED attribute.

FS

Indicates that Fujitsu standard shift codes (28,29) are to be used for mixed data and that new character fields are to be defaulted to have the SBCS attribute.

IM

Indicates that IBM standard shift codes (0E,0F) are to be used for mixed data and that new character fields are to be defaulted to have the MIXED attribute.

IS

Indicates that IBM standard shift codes (0E,0F) are to be used for mixed data and that new character fields are to be defaulted to have the SBCS attribute.

OFF

Indicates that the DBCS features are to be disabled.

Valid Entries:

FM, FS, IM, IS, or OFF

Default Value:

(No default)

Example JCL

The following shows the command to activate a double-byte character set.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          CXXMAINT   OPTION=ALTER,DBCS=IM
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT ALTER DBCS

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
         ❶          CXXMAINT OPTION=ALTER,DBCS=IM

                    FUNCTION=CXXMAINT
         ❷             DBCS=IM




         ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹         *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT ALTER DSN (Modify Data Set Name)

When to Use

During the INIT function, CA Datacom/DB saves the data set name (DSN) for Index or data areas in the Directory. Use the CXXMAINT ALTER DSN option to modify the DSN for reporting purposes and, optionally, to change the DSN used for dynamic allocation. This eliminates having to reinitialize the Index or data areas to change the DSN in the Directory.

You can use this option with a database open in a Multi-User Facility.

How to Use

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:

```
▶▶─ CXXMAINT OPTION=ALTER,AREA=aaa ─ ,DBID=n,DSN=dataset ──────────────────────▶◀
```

Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,AREA=

Identifies the data or Index Area whose data set name is to be changed.

Valid Entries:

The DATACOM-NAME of the data area, or the index area name (IXX or I*nn*, where *nn* is 00 to 99).

Default Value:

(No default)

,DBID=

Identifies the database associated with the data or Index Area.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,DSN=

Specifies the new data set name.

The DSN= cannot be set to spaces. A space serves as a delimiter and terminates the name. Do not use quotation marks to enclose the name.

Note: See restrictions on changing data set names below.

Valid Entries:

Valid data set name

Default Value:

(No default)

Note: The following are special considerations to be aware of.

1. The data set name for the base used for the CBS index can be changed only in one of the following ways:
   - The enabled Multi-User Facility has a different database as the CBS index.
   - The Multi-User Facility is not enabled, is not externally secured for DBUTLTY, and the DBUTLTY INIT function is used, as in the following example:

```
DBUTLTY INIT DBID=6,AREA=IXX
DBUTLTY INIT DBID=17,AREA=IXX
DBUTLTY INIT DBID=17,AREA=TTM
DBUTLTY LOAD DBID=17,FORMAT=NONE,KBYTES=9999,SORT=1
```

2. The data set name for the SQL temporary table management (TTM) index (IXX) or data area (TTM) may only be changed in one of the following manners:
   - The enabled Multi-User Facility has a virtual TTM base.
   - The enabled Multi-User Facility has SQL not generated.
   - The Multi-User Facility is not enabled, is not externally secured for DBUTLTY, and the DBUTLTY INIT function is used.
3. A data area or index area data set cannot be shared by two CXXs. This can result in data integrity problems and index corruption.

Example JCL

The following shows the command to alter the data set name for the data area DEM in database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
       CXXMAINT  OPTION=ALTER,AREA=DEM,DBID=1,DSN=DASUP.ALPHA.DEM001
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT ALTER DSN

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          CXXMAINT OPTION=ALTER,AREA=DEM,DBID=1,DSN=DASUP.ALPHA.DEM001

                    FUNCTION=CXXMAINT
                       AREA=DEM
          ❷            DBID=1
                       DSN=DASUP.ALPHA.DEM001
                       OPTION=ALTER

          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹         *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT ALTER DSOP (Modify Space Reclamation Option)

When to Use

Use this option to change the space reclamation option specified for an area. The new space reclamation option is used immediately.

You can use this option with a database open in a Multi-User Facility.

Index Area Organization

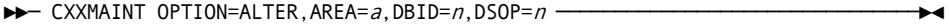
The ALTER DSOP option does not update the Index Area (IXX). The old option is still in effect for existing entries.

Synchronization with CA Datacom/DB

You must also change the space management option attribute (SPACE-MNGMNT) for the area in CA Datacom/DB. Otherwise, the old option is restored if the CA Datacom/DB definition is subsequently cataloged to the CA Datacom/DB Directory.

How to Use

Multi-User mode is required to execute this command. To change the space reclamation option for an area, execute DBUTLTY with the command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,AREA=

Identifies the area for which the space management option is to be altered.

Valid Entries:

DATACOM-NAME of the area

Default Value:

(No default)

,DBID=

Identifies the database associated with the data area.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,DSOP=

Specifies the new space reclamation option.

0

No space reclamation

1

Basic space reclamation

2

Wraparound

3

Clustering

Valid Entries:

0, 1, 2, or 3

Default Value:

(No default)

Example JCL

The following shows the command to change the space reclamation option for the PAY area to no space reclamation (option 0).

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=ALTER,AREA=PAY,DBID=1,DSOP=0
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT ALTER DSOP

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .....1.....2.....3.....4.....5.....6.....7.....8
          ❶          CXXMAINT OPTION=ALTER,AREA=PAY,DBID=001,DSOP=0

                    FUNCTION=CXXMAINT
                       AREA=PAY
          ❷            DBID=1
                       DSOP=0
                       OPTION=ALTER

          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹           *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT ALTER LANGUAGE (Activating LANGUAGE Support)

The ALTER LANGUAGE option allows you to specify whether to use the Katakana language support. Also see CXXMAINT ALTER DBCS (Activating DBCS Support) (see page 209).

Note: If you set your environment for DBCS or Katakana data, entity-occurrence names and data that contains DBCS or Katakana data may be displayed on a terminal in CA Datacom/DB and CA Dataquery. However, if a terminal does not support the display of DBCS or Katakana data, the terminal may display the data incorrectly or a terminal error such as a PROG402 abend can occur.

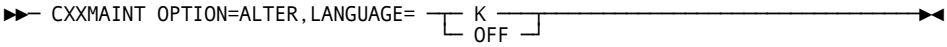
When to Use

Use the ALTER LANGUAGE option to turn on Katakana language support or to disable the LANGUAGE feature.

How to Use

The MUF must be active when you execute this command. To activate LANGUAGE support, use the following command:

Note: If you are using SQL, you must restart the MUF after running this command.



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,LANGUAGE=

Indicates whether to use Katakana language support.

K

Indicates Katakana language support is provided.

OFF

Indicates that the LANGUAGE feature is to be disabled.

Valid Entries:

K or OFF

Default Value:

(No default)

Example JCL

The following shows the command to activate the LANGUAGE feature.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see .
// ASSGN SYSnnn,device-type,VOL=volser,SHR
// DLBL CXX,'cxx',,DA                              Directory data set
// EXTENT SYSnnn,volser
// EXEC DBUTLTY,SIZE=750K
         CXXMAINT   OPTION=ALTER,LANGUAGE=K
/*
/&
* $$ EOJ
```

Sample Report: CXXMAINT ALTER LANGUAGE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶         CXXMAINT OPTION=ALTER,LANGUAGE=K

                    FUNCTION=CXXMAINT
          ❷           LANGUAGE=K




          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹         *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT ALTER LINK (Disable CXX Name Edit)

The ALTER LINK option allows you to suppress an edit to ensure that an Index Area (IXX) or data area is being accessed through the proper Directory (CXX). With regard to using the ALTER LINK option, see the caution below.

When to Use

Most user environments include from a few up to dozens of separate CA Datacom/DB systems in such categories as production, test, and QA. Each environment is expected to comprise a complete set of information with a Directory that has a unique name.

Because all data set protection is provided through the Directory, each Directory having a unique name is an important factor in clearly knowing which data sets are in TEST status and which are in PRODuction status. The same holds true if, for example, PRODuction Europe is separate from PRODuction Africa. All systems have private copies of data sets within the system. Therefore, all updating is controlled through one Directory (CXX) and one copy of the CA Datacom/DB code.

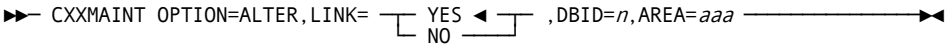
Some sites have a very large unchanging *reference* type database and want to have the same physical copy of it accessed by all of the various Multi-User Facility systems. We do not recommend this. It can only be done safely if you follow these rules:

- It should not be changed by any system unless closed to all except the one doing the change. *All protection is the responsibility of the user. CA Datacom/DB cannot detect this danger or protect against it.*
- No utility should process against the data set if it is open by any other utility or the Multi-User Facility. *All protection is the responsibility of the user.*
- At the database level, the Index Area and each data area needs to have this option (with LINK=NO) specified to disable editing the Directory name, that is, to prevent it from being changed upon initialization to the Directory name in the data set.

Important! ALTER LINK is rarely used, but if you do use this option, do so with extreme caution. If the above rules are violated, the data and Index Area can be corrupted with CA Datacom/DB being unaware of the corruption and unable to correct it.

How to Use

Multi-User mode is required to execute this command. Use the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,AREA=

Identifies the data or index area.

Valid Entries:

The DATACOM-NAME of the data area or index area name (IXX or I*nn*, where *nn* is 00 to 99).

Default Value:

(No default)

,DBID=

Identifies the database associated with the Index Area or data area(s).

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,LINK=

Allows you to suppress (LINK=NO) an edit of the Directory name during an open in order to ensure that the Index Area and data area(s) associated with a particular database are being accessed through the proper Directory (CXX).

NO

Disables the Index Area or data area editing of the Directory name. With regard to using LINK=NO, see the caution in CXXMAINT ALTER LINK (Disable CXX Name Edit) (see page 221).

YES

Allows the editing of the Directory name.

Valid Entries:

YES or NO

Default Value:

YES

Example JCL

The following shows the command to suppress an edit of the CXX name for an area in database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=ALTER,DBID=1,LINK=NO,AREA=PAY
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT ALTER LINK

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶           CXXMAINT OPTION=ALTER,DBID=1,LINK=NO,AREA=PAY

                    FUNCTION=CXXMAINT
                       AREA=PAY
                       DBID=1
          ❷              LINK=NO
                       OPTION=ALTER

          ❸.
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹           *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT ALTER LOGGING (Start/Stop Logging by Table)

When to Use

Use ALTER LOGGING to activate or deactivate maintenance logging for any table that is defined in CA Datacom/DB with RECOVERY=Y. Deactivating logging can be useful when testing new applications. Logging can be activated when a new application is placed into production.

If you deactivate logging for a table, you must provide your own method of ensuring data integrity, since CA Datacom/DB has no record of maintenance activity for that table without the logging facility.

Active Processing

You can use this option with a database open in a Multi-User Facility. Or, you can use ACCESS (Restricting Opens or Maintenance) (see page 55) to prevent opens of the database containing a table that you want to change, with the instruction to wait until current tasks complete.

Impact on Recovery

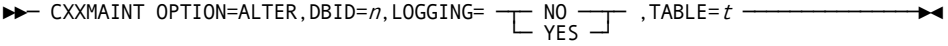
Carefully consider the consequences of turning logging on or off. Logging directly relates to CA Datacom/DB Recovery features, including transaction backout. See the *CA Datacom/DB Database and System* Administration *Guide* for details about logging.

We recommend that logging be active for any table containing production data. Logging is an integral part of the Pipeline performance option that significantly reduces the overhead to perform maintenance operations. See the *CA Datacom/DB Database and System Administration Guide* for details about the Pipeline option.

Proper execution of SQL statements that cause data change requires all affected tables to have logging enabled. Logging should not be turned off (LOGGING=NO) for any SQL maintenance. This applies to both the CA Datacom/AD definition and the DBUTLTY CXXMAINT option.

How to Use

Multi-User mode is required to execute this command. Execute the DBUTLTY with the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,DBID=

Identifies the database containing the table for which logging is to be activated or deactivated.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,LOGGING=

Indicates the action for logging. NO disables logging and YES activates logging.

Valid Entries:

NO or YES

Default Value:

(No default)

,TABLE=

Identifies the table for which logging status is to change.

Valid Entries:

DATACOM-NAME of the table that resides in the database specified in the DBID parameter

Default Value:

(No default)

Example JCL

The following shows the command to deactivate logging for the table named POL in database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=ALTER,DBID=1,LOGGING=NO,TABLE=POL
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT ALTER LOGGING

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶           CXXMAINT OPTION=ALTER,DBID=001,LOGGING=NO,TABLE=POL

                    FUNCTION=CXXMAINT
                       DBID=1
          ❷              LOGGING=NO
                       OPTION=ALTER
                       TABLE=POL

          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹         *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT ALTER OPTION2= (Activating Other Features)

The CXXMAINT OPTION=ALTER function uses the OPTION2= keyword to activate other features as follows:

- OPTION2=*datafs* specifies DATA FAST SEARCH (DATAFS) option (see CXXMAINT ALTER OPTION2=datafs (DATAFS Support)).
- OPTION2=*datahu* specifies the high used mark option (see CXXMAINT ALTER OPTION2=datahu (High Used Mark)).
- OTION2=*areaev* specifies the data area data set extent validation option (see CXXMAINT ALTER OPTION2=areaev (Data Area Extent Validation)).

CXXMAINT ALTER OPTION2=datafs (DATAFS Support)

A feature that reduces compression overhead and provides an optional search algorithm named DATAFS (DATA FAST SEARCH). The DATAFS algorithm provides possible performance-enhanced searching of multi-table areas and areas with tables that use compression.

When to Use

Use the ALTER DATAFS option when you want to enable the DATA FAST SEARCH feature.

Prior to Version 12.0, each data area that contained a single table that was not compressed was searched for a record using a binary search. The ability to do the binary seach was turned on at data load time if a physical block could contain 15 or more data rows. Otherwise, the blocks were searched in a sequential physical order. For large block sizes with small records, such a sequential search could become a significant percentage of MUF CPU usage. Using DATAFS, however, the last 16 bytes of each data block is used by the search algorithm to provide a faster method of finding individual data rows, especially for large block and small row sizes. Once created, these 16 end bytes are stored on DASD with the data.

Migration Considerations

Because using the feature that reduces compression overhead physically alters the data area, there are two fallback considerations to consider, that is, you must consider what happens if:

- You run the previous release (Version 11.0, also known as r11) against the area after implementing this Version 12.0 feature.
- The global option is turned off after implementing this feature.

In both cases, reading and searching the data block is not affected because each data block has stored within it the available space in the block. When searching the block, that information defines when the logical end of the block is reached. The available space includes the 16 bytes used by the DATAFS search algorithm. It is therefore completely transparent to the old block search code. Even if you fall back to the previous release, unless maintenance is done to a row within the block the 16 bytes remain at the end of the block.

If a data block is changed due to a maintenance command after the option has been turned off, the maintenance clears the last 16 bytes, if it has been used by DATAFS. A previous release fallback solution that does the same thing is provided.

The remaining concern for the previous release is to consider what would happen if the fallback solution has not been implemented at a site which falls back after setting the global DATAFS option.

As each data area data set is opened, the DATAFS option is checked. If it is YES and it has not already been done for this data area, it is marked as incompatible with the previous release (unless the fallback solution is in place). Future previous release opens of this data set fail with a return code 72 (003) if the data area has been thus marked and if the previous release fallback solution is not in place.

The DBUTLTY INIT of a data area also marks the data area, no matter the DATAFS setting.

Reporting DATAFS in Full CXX Report

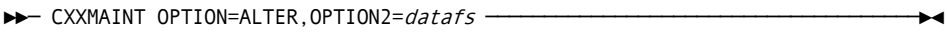
A full CXX report includes information on DATA FAST SEARCH *status*, where *status* in the report is either YES or NO.

Security Considerations

This functions is secured in the DTUTIL resource class with a resource name ending in CXXMAINT.ALTEROP2. See the DBUTLTY Resource List section of the *CA Datacom Security Reference Guide* for details. No underlying table rights are checked.

How to Use

To use the DATA FAST SEARCH option, use the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,OPTION2=*datafs*

Indicates whether to use the DATAFS search algoritm for performance enhanced searching of multi-table areas and areas with tables that use compression.

Specify *datafs* as N_DATAFS when you do not want to use DATAFS.

Specify *datafs* as Y_DATAFS when you want to use DATAFS. When the DATAFS feature has been turned on, each data block in a Unique Row Identifier (URI) area is eligible to use the new algorithm that includes altering the DASD for a given block. A URI is a sequential number assigned when a row is added or inserted that CA Datacom/DB uses to identify a row. Areas which qualified for using the binary search algorithm in previous releases continue to use the previous algorithm. Each data block uses the new DATAFS approach if there are a sufficient number of rows in the block to make the new approach worthwhile.

Once the global DATAFS option is turned on, blocks have the new search criteria built in one of two ways.

- The search criteria is built and stored at the end of the data block if, as rows are added to new or existing blocks, the search criteria has not yet been built, there are a sufficient number of rows in the block, and the block currently has at least 16 bytes of space available at the end.
- The DBUTLTY function RETIX,AREA=*DATA builds the search criteria if DATAFS is set to Y for URI data areas that are not using the old style binary search. Since this DBUTLTY already runs in MUF, it allows sites with existing populated URI data areas to build the search criteria in MUF (and on DASD) without taking an area offline. For each individual data block, however, the search criteria can be built only if there are 16 bytes available at the end of the block.

Because the new DATAFS algorithm needs the last 16 bytes of each data block, it is possible for each existing data area to require more physical DASD space. This includes all methods of newly populating a data area (LOAD, and so on), or inserting rows in an existing data area. There are two exceptions: RECOVERY and LOAD from a RECID backup using data from before DATAFS is set to Y. In these two cases, physical data placement takes precedence over the 16 bytes used by DATAFS.

Valid Entries:

Y_DATAFS or N_DATAFS

Default Value:

(No default)

CXXMAINT ALTER OPTION2=datahu (High Used Mark)

Beginning with Version 12.0, OPTION2=Y_DATAHU/N_DATAHU replaces DIAGOPTION CXX,08,ON/OFF from the previous release.

When to Use

Use this option when you want to turn on or turn off the "high used mark" option.

When this option is not turned on, each execution of a DBUTLTY LOAD loads the data (if any) then formats the rest of the data set by writing binary zero blocks. It is possible, especially at sites using DASD mirroring, for these formatting writes to overwhelm the system, thereby causing possible performance degradation. Also, without this option turned on, physical retrieval against an area, such as the GETPS command and, for example, the DBUTLTY function BACKUP SEQ=PHYSICAL, requires the entire data set be read even though much of it may never have been used.

After reading the following information, you can decide if this feature is desired at your site. If so, you can turn it on for all data sets in the Directory (CXX). Once turned on, some existing programs and DBUTLTY jobs might execute in less time, because thereafter less than the whole data set area is loaded and (or) read. It is therefore possible that significant performance gains can be realized for sites that frequently load their data or use the GETPS command in programs.

After this option is turned on, the first execution of the DBUTLTY function LOAD after a DBUTLTY function INIT of an area continues to function as it did before this option was turned on. All subsequent LOADS, however, including REORG, and physical retrievals (GSETP command and BACKUP SEQ=PHYSICAL) rely on a special form of the "next available overflow" called the high used mark feature.

The next available overflow is a relative DASD location, part of the data set that has never been actively used by CA Datacom. The high used mark represents the point up to which LOAD must format the data set and up to which physical retrieval must read. It is the next available overflow number rounded up to 64 tracks.

An INIT of a data area sets the high used mark to zero. A value of zero means LOAD must format the entire data set. At the start of LOAD, the current high used mark is saved and the DASD version is set to zero. If the data LOAD takes less space than the saved copy of the high used mark, the data set is formatted up to that mark. After the data portion of the LOAD is finished but before the index portion of the LOAD is done, the DASD high used mark is updated to indicate the point to which "true data" has been loaded, rounded up to a 64-track boundary. If the LOAD fails during the data portion, the next LOAD must format the entire data set. If the LOAD fails during the index portion, the high used mark is used for formatting.

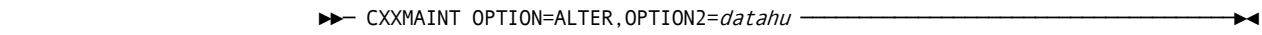
The runtime code in the Multi-User Facility that maintains the next available overflow notices each time the next available overflow number crosses a 64-track boundary. If the memory copy of the high used mark is non-zero, Multi-User Facility reads the data set control block. It then updates the high used mark and immediately writes the data control block.

A GETPS command treats the high used mark as the end of the data set unless the area is not opened for update. If it is not open for update, GETPS reads to the true end of the data set. This is done for the sake of read-only Multi-User Facilities. LOAD always treats the high used mark as the end of the data set.

If you turn off this option, it reverses the marking of the CXX caused by turning the option on. When an area is opened for update, its compatibility level is checked. If it is level one and the CXX has been marked to stop using the high used mark feature, the compatibility level is reset to 0, the high used mark is zeroed.

How to Use

To turn the high used mark option on or off, use the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,OPTION2=*datahu*

Sets the "high used mark" feature.

Specify *datahu* as N_DATAHU when you do not want to turn on the high used mark feature.

Specify *datahu* as Y_DATAHU when you want to turn on the data high used mark feature.

Valid Entries:

Y_DATAHU or N_DATAHU

Default Value:

(No default)

CXXMAINT ALTER OPTION2=areaev (Data Area Extent Validation)

Beginning with Version 12.0, OPTION2=Y_AREAEV/N_AREAEV replaces DIAGOPTION CXX,32,ON/OFF from the previous release.

When to Use

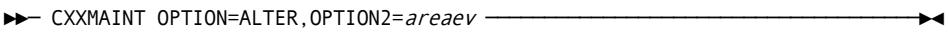
Use this option when you want to turn on or turn off data area data set extent validation.

CA Datacom/DB uses data set extent validation in the following way. When a data area is not open, users can and sometimes do move the current extents to more extents, less extents, or different volumes. The data set may even be uncataloged and then recataloged to add volumes. All tracks of the data set would not be correctly preserved if a mishap should occur during the moving of current extents. CA Datacom provides a certain amount of protection to avoid the data corruption such mishaps could cause. This option provides even more protection by performing the following steps.

After the open of a data area, the control block is read and verified. The last block on the last track of every extent is then read. This read validates that the block exists and is of the proper size. It is not, however, validated to look proper in content. This option also places a control block as the last block of the data set. It is built during the LOAD or REORG load and moved during an EXTEND or dynamic extend. If this option is turned on, the open process verifies it is correct.

How to Use

To turn data area data set extent validation on or off, use the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,OPTION2=*areaev*

Used for turning on and off "data area data set extent validation".

Specify *areaev* as N_AREAEV when you do not want to turn on data area data set extent validation.

Specify *areaev* as Y_AREAEV when you want to turn on data area data set extent validation.

Valid Entries:

Y_AREAEV or N_AREAEV

Default Value:

(No default)

CXXMAINT ALTER OPTION2=sumode (Single User Mode)

By default, when you convert a Version 11.0 (also known as r11) CXX to a Version 12.0 CXX, use of Single User mode is blocked, even if secured Single User is allowed.

When to Use

Use this option when you want to run Single User using the OPTION2=Y_SU option with CXXMAINT or to remove support for Single User using OPTION2=N_SU. This option is provided to the full CXX report and has been added to the DB00271I MUF startup message (for details about DB00271I, see the *CA Datacom/DB Message Reference Guide*).

The new Single User option is provided to the full CXX report as follows:

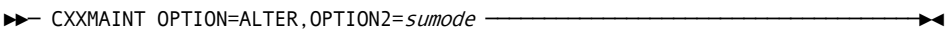
```
SECURE USING JOBNAME - NO               CXX LEVEL - 1
SECURE ALLOWING SINGLE USER - NO        SINGLE USER ALLOWED - NO
```

We have long recommended against using Single User mode. The Single User environment allows no logging and therefore has no restart or recovery capability. In addition, running in Single User bypasses external security. However, CA Datacom continues to support using Single User mode for the DBUTLTY functions that run Single User and the DBFLSUB feature that loads an area with user program input. Prior to Version 12.0, only if the CXX was externally secured did you have the option of using DBUTLTY CXXMAINT OPTION=ALTER,SINGLE=YES to allow Single User mode, or SINGLE=NO to block Single User mode.

If a Single User open that is allowed occurs, message DB00138I is issued, confirming that fact and documenting the Single User execution (for details om DB00138I, see the *CA Datacom/DB Message Reference Guide*).

How to Use

To turn Single User mode on or off, use the following command.



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,OPTION2=*sumode*

Use the *sumode* variable to turn Single User mode on by specifying Y_SU for *sumode* or to turn Single User support off by specifying N_SU for *sumode.*

Valid Entries:

Y_SU or N_SU

Default Value:

(No default)

CXXMAINT ALTER SINGLE (Run in a Single User Environment)

This option allows you to run in a Single User environment with external security. At installation, the system is set to run in a Multi-User environment.

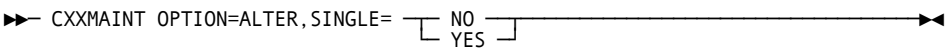
When to Use

Use ALTER SINGLE to execute Single User jobs with external security or, after executing this option, to run in a Multi-User environment. If you use this option, Single User jobs run *unsecured* and all Multi-User jobs run secured.

Caution We strongly recommend that Single User jobs not be allowed to run against a Directory (CXX) that is externally secured. To ensure this, Security Administrators should allow no users access to the DTUTIL resource cxxname.DBUTLTY.CXXMAINT.ALTERSIN. If Single User is allowed to run against an externally secured Directory, table-level access for Single User is unsecured, that is to say, any Single User job can update any table with no security check.

How to Use

To run Single User jobs, execute DBUTLTY with the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

SINGLE=

Indicates if you are running in a Single User environment or a Multi-User environment.

Valid Entries:

NO or YES

Default Value:

(No default)

Example JCL

The following shows the command to allow Single User jobs to execute while you are using external security.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=ALTER,SINGLE=YES
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT ALTER SINGLE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶           CXXMAINT OPTION=ALTER,SINGLE=YES

                    FUNCTION=CXXMAINT
          ❷              OPTION=ALTER
                       SINGLE=YES


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹         *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT CONVERT1112 (Upgrade CXX to Current Version)

The CXXMAINT CONVERT1112 option of DBUTLTY upgrades your CA Datacom/DB Directory (CXX) from the previous version to make it compatible with the installed new version. The CONVERT1112 option reads the Directory backup from a prior version, alters it, and writes the Directory in the new format. See the *CA Datacom Installation and Maintenance Guide* for details about upgrading.

When to Use

Use CONVERT1112 only when upgrading the Directory from one version to another. This option is *not* used to upgrade to a new service pack level. Also, this option does not allow you to skip a version. You must convert to a new version from the version immediately preceding.

Backup Requirements

Since this option only reads data in the format produced by a backup, you must have a backup of your Directory to run the CONVERT1112 option. Create the backup using the BACKUP function of the old version.

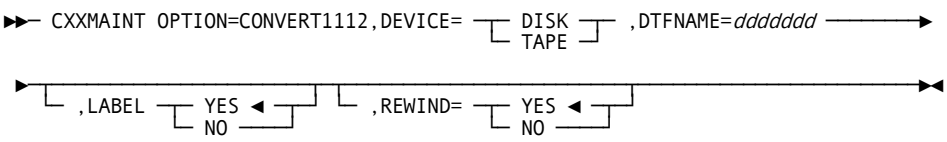
The backup that you use as input data for the CONVERT1112 option must be current, since all data in the Directory is replaced by data converted from the backup. The new Directory can be a different data set from the one from which the backup was made.

Directory INIT Requirements

This option must follow an INIT CXX function for the new Directory in the same utility execution or as the next execution. CA Datacom/DB retains the Directory name from the INIT, but builds all other information from the backup.

How to Use

Multi-User mode is required to execute this command. Execute DBUTLTY using the command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=CONVERT1112

Requests the CXXMAINT CONVERT1112 option.

DEVICE=

Specifies the device type for the input file.

When you specify DEVICE=TAPE, the system logical unit (SYS) number must be 5, (SYS005).

Valid Entries:

DISK or TAPE (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the input file. If DEVICE=TAPE, this name must match the name which you code in your z/VSE JCL TLBL statement, otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

The name required to access the backup data set

Default Value:

(No default)

Optional Keywords

LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), LABEL= specifies whether tape labels are to be processed. We recommended that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates that the tape is to be rewound before the OPEN and after the CLOSE. NO indicates no rewind before the OPEN or after the CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

Example JCL

The following shows the command to upgrade your CA Datacom/DB Directory for compatibility with the installed new version, using the backup created under the previous version as input. In the following example, the input is called CXXIN.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL CXXIN,'cxx.bkup'                           Previous version CXX backup
 // EXEC DBUTLTY,SIZE=750K
          CXXMAINT   OPTION=CONVERT1112,DEVICE=TAPE,DTFNAME=CXXIN
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT CONVERT1112

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                             General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶         CXXMAINT OPTION=CONVERT1112,DEVICE=TAPE,DTFNAME=CXXIN

                    FUNCTION=CXXMAINT
          ❷           DEVICE=TAPE
                     DTFNAME=CXXIN
                     OPTION=CONVERT1112

          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹           *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT CONVERT1211 (Downgrade CXX to Prior Version)

The CXXMAINT CONVERT1211 option of DBUTLTY downgrades your CA Datacom/DB Directory (CXX) from the current version to the prior version. The CONVERT1211 option reads the Directory backup, alters it, and writes the Directory in the old format.

This is the only DBUTLTY Version 12.0 execution allowed while pointing to a CXX set to the previous release (Version 11.0, also known as r11).

When to Use

Use CONVERT1211 only when downgrading (falling back) the Directory from the current version to the prior version.

Backup Requirements

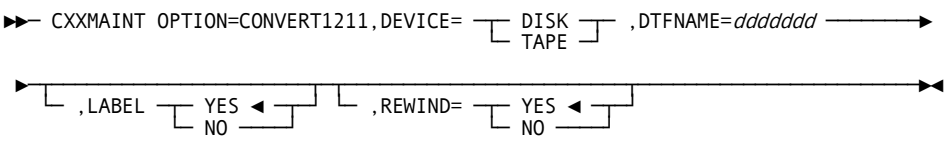
The backup must be taken with the current release.

Directory INIT Requirements

This option must follow an INIT CXX function for the new Directory execution with the prior release, the next step. The Directory name is retained from the INIT but builds all other information from the backup.

How to Use

Multi-User mode is required to execute this command. Execute DBUTLTY using the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=CONVERT1211

Requests the CXXMAINT CONVERT1211 option.

DEVICE=

Specifies the device type for the input file.

When you specify DEVICE=TAPE, the system logical unit (SYS) number must be 5, (SYS005).

Valid Entries:

DISK or TAPE (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the input file. If DEVICE=TAPE, this name must match the name which you code in your z/VSE JCL TLBL statement, otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

The name required to access the backup data set

Default Value:

(No default)

Optional Keywords

LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), LABEL= specifies whether tape labels are to be processed. We recommended that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates that the tape is to be rewound before the OPEN and after the CLOSE. NO indicates no rewind before the OPEN or after the CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

Example JCL

The following shows the command to downwgrade your CA Datacom/DB Directory to prior version, using the backup created using the current version as input. In the following example, the input is called CXXIN.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL CXXIN,'cxx.bkup'                           Previous version CXX backup
 // EXEC DBUTLTY,SIZE=750K
         CXXMAINT   OPTION=CONVERT1211,DEVICE=TAPE,DTFNAME=CXXIN
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT CONVERT1211

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                             General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          CXXMAINT OPTION=CONVERT1211,DEVICE=TAPE,DTFNAME=CXXIN

                    FUNCTION=CXXMAINT
          ❷            DEVICE=TAPE
                      DTFNAME=CXXIN
                      OPTION=CONVERT1211

          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

          ❹           *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT DDPROD (Update Directory)

The CXXMAINT DDPROD option updates the Directory (CXX) with CA Datacom/DB definitions. This option reads the input file as created by CA Datacom/DB and updates the Directory to reflect any changes, additions, or deletions.

CA Datacom/DB also provides a means of cataloging CA Datacom/DB definitions directly to the CA Datacom/DB Directory. For details on the -CXX CATALOG and 1000 CATALOG batch transactions, see the *CA Datacom Datadictionary Batch Reference Guide*. For information about the online CATALOG function, see the *CA Datacom Datadictionary Online Reference Guide* or the *CA Datacom Datadictionary User Guide*.

When to Use

Use the DDPROD option to update the Directory when adding or changing definitions in CA Datacom/DB which must be reflected in the Directory. Perform the tasks in the following order:

1. Define the new table or modify the existing table definition in CA Datacom/DB.
2. Execute the CA Datacom/DB DDCFBLD utility to create the input for the DDPROD option. (See the *CA Datacom Datadictionary Batch Reference Guide* for details.)
3. Use the DDPROD option to update the Directory to reflect the CA Datacom/DB definitions.

You can also use the DDPROD option to create a new database with a different database identifier from an existing production database definition. This option is useful to create a replica of a database for testing purposes.

Important! Do **not** use the CXXMAINT DDPROD option for SQL accessible tables, or on a database containing SQL accessible tables, or on databases containing partitioned tables. Instead, use the CA Datacom/DB CATALOG transaction.

Active Processing

You can change an existing database definition as long as no task is actively processing the database. Use ACCESS (Restricting Opens or Maintenance) (see page 55) to prevent opens to a database that you want to change.

Processing a DDPROD Request

As CA Datacom/DB processes each table definition, the current Directory content is compared to the CA Datacom/DB definitions.

If any tables in the database being updated are not loaded, they are deleted from the Directory at the start of the process. A report of the Directory taken at this time does not provide area space information. The next LOAD of the area restores the information.

The action taken by CA Datacom/DB depends on the situation encountered, as follows:

New Area, New Table

If an area is new and no match exists in the Directory, CA Datacom/DB adds the area with no initialization information. CA Datacom/DB adds the new table with no initialization information, and sets it as not loaded.

Existing Area, New Table

If the table is new, CA Datacom/DB adds it. If the area containing the table exists and is loaded, CA Datacom/DB sets the new table as loaded and having zero records.

If you are adding a table to an existing area, CA Datacom/DB updates the area with the new CA Datacom/DB entity-occurrence information, data set space option, and slack information to the new values. If the block size differs, CA Datacom/DB saves the new block size in the area definition as the next block size until the next INIT of the data area.

Existing Area, Existing Table

If the table currently exists, CA Datacom/DB compares the Directory and CA Datacom/DB versions. If a critical change has occurred, CA Datacom/DB leaves the current table in *HISTORY* status and adds the new table as not loaded.

Special codes on the Impact Report indicate critical changes and the reason. The codes and reasons associated with an Impact Report are as shown in the following list.

Impact Report Codes and Reasons

00

NOT LOADED (00) in the Impact Report means that a table was NOT LOADED for reasons other than a CATALOG or DDPROD. This indicates that the area has been loaded since the last DDPROD or CATALOG but is not loaded now. The *current* NOT LOADED status is of no importance with regard to the last DDPROD or CATALOG. For information about requesting an Impact Report as a DBUTLTY function independent of a DDPROD, that is, a TYPE=I CXX report, see Type I - Impact Report (see page 527).

01

New table name

02

Area name change

08

Table ID change

09

Recover yes/no change

10

CA Datacom/DB compression or user compression change

11

Record length change

13

User compression name change

14

Special field processing change

- When you have to reload the data area to make the new table definition usable, you can run a SEQ=NATIVE BACKUP or EXTRACT function which use the *HISTORY* definition to access the old data, change its format if necessary, and reload the area. When the reload completes, CA Datacom/DB marks the new table definition loaded.

  Caution If you have changed table definitions in multiple areas in the database, be sure that you back up all affected areas before reloading any data. The first reload of **any** data area in the database deletes all *HISTORY* database definitions. If you have not unloaded all changed areas before doing a reload, the *HISTORY* definition is no longer available.

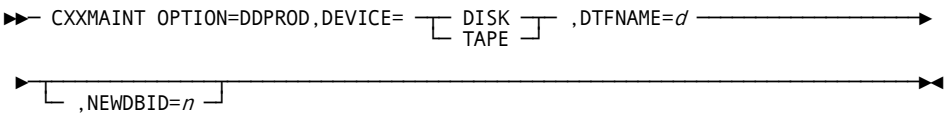
  If you are moving a table from one area to another, back up the area or extract the table before cataloging or using DDPROD to update the definition.

  Do **not** use the CXXMAINT DDPROD option for SQL accessible tables or on a database containing SQL accessible tables. Instead, use the CA Datacom/DB CATALOG transaction.

- If a key change occurs, CA Datacom/DB removes the *HISTORY* version, adds the new table with the data loaded but the Index is marked as not loaded. (Run the RETIX function to correct the Index.)
- If neither of these situations occurs, CA Datacom/DB regards the changes as noncritical, removes the *HISTORY* version, and adds the new table as loaded.

How to Use

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=DDPROD

Requests the option to update the CXX from CA Datacom/AD definitions.

DEVICE=

Specifies the device type for the input file.

When you specify DEVICE=TAPE, the system logical unit (SYS) number must be 5, (SYS005).

Valid Entries:

DISK or TAPE (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the input file. If DEVICE=TAPE, this name must match the name which you code in your z/VSE JCL TLBL statement, otherwise, this name must match the name you code in your JCL DLBL statement.

Valid Entries:

The name required to access the data set created by the DDCFBLD utility

Default Value:

(No default)

Optional Keyword

,NEWDBID=

Specifies a different database ID to be assigned to the definition. Use this keyword to create a set of new definitions quickly that are identical to a production base. Do *not* use this keyword when performing a normal update.

Note: Do not use this parameter for databases accessed by SQL or CA Dataquery. These products must have the correct DBID to reference additional information from CA Datacom/DB about the production database. If the NEWDBID= parameter is used for a database accessed by CA Ideal, the corresponding ASSIGN DBID or ALTER DBID command must be run before running the application. See CA Ideal documentation for details.

Valid Entries:

DATACOM-ID of the database

Default Value:

When this keyword is not coded, the database is updated as indicated from the input.

Example JCL

The following shows the command to update the CA Datacom/DB Directory, using the CA Datacom/DB data in DDCFBLD.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL DDCFBLD,'dd.output'                        Input from CA Datacom/AD
 // EXEC DBUTLTY,SIZE=750K
          CXXMAINT   OPTION=DDPROD,DEVICE=TAPE,DTFNAME=DDCFBLD
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT DDPROD

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
      ❶              CXXMAINT OPTION=DDPROD,DTFNAME=DDCFBLD,DEVICE=TAPE

                    FUNCTION=CXXMAINT
      ❷                DEVICE=TAPE
                      DTFNAME=DDCFBLD
                      OPTION=DDPROD

      ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                             General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                                                IMPACT REPORT

                    BASE        AREA        TABLE ID        DISPOSITION
                                                                                                  ❹
                    390         ALI         ALI  003        LOADED
                                ALT         ALT  001        OLD
                                            ALT  001        NOT LOADED  (11)
                                APL         APL  002        LOADED
                                LLN         LLN  004        LOADED
                                MST         MST  005        INDEX NOT LOADED
                                SCR         SCR  006        LOADED
```

❹

DBUTLTY reports the disposition of each table in the database as one of the following:

LOADED

The definition for this table supplied in the input from DDCFBLD contained only noncritical changes from the definition for this table previously in the Directory.

OLD

The definition for this table supplied in the input from DDCFBLD contained critical changes from the definition for this table previously in the Directory. The OLD definition is the definition for this table as it existed prior to the execution of the CXXMAINT DDPROD.

NOT LOADED

The NOT LOADED definition is the new definition for this table. The numeric code in parentheses indicates the reason CA Datacom/DB marked this table as not loaded. See the chart on page Processing a DDPROD Request (see page 247) for an explanation of the possible codes. Use the DBUTLTY LOAD function to reload this table. When the LOAD completes, DBUTLTY marks this table LOADED and deletes the old definition.

INDEX NOT LOADED

The definition for this table supplied in the input from DDCFBLD contained one or more key changes from the definition for this table in the Directory. Use the DBUTLTY RETIX function to correct the index.

CXXMAINT DELETE (Delete CXX Definitions)

When to Use

The CXXMAINT DELETE option deletes table and database definitions from your Directory (CXX).

Deleting a Database

Use the DELETE option to remove an entire database. If any tables still exist in the database to be deleted, those tables are deleted. Prior to Version 11.0 if any tables were left in a database, the delete of that database would fail, but beginning with Version 11.0, it is no longer necessary to delete table definitions in a database separately before deleting that database, even though it is still possible to do so.

To remove a database using CXXMAINT DELETE, the database must not currently be open. If the database is open with no users, close the database, then issue the delete.

Note: A virtual database that is open cannot be closed, and therefore it cannot be deleted.

Deleting Tables

If any tables still exist in the database to be deleted, those tables are deleted. Prior to Version 11.0 if any tables were left in a database, the delete of that database would fail, but beginning with Version 11.0, it is no longer necessary to delete table definitions in a database separately before deleting that database, even though it is still possible to do so.

Use the following to delete one or more tables within a database, but not the entire database.

Non-SQL Accessible Tables

See the *CA Datacom Datadictionary Batch Reference Guide* or *CA Datacom Datadictionary Online Reference Guide* for instructions on accomplishing the following. This procedure ensures that the Directory and the CA Datacom/DB definitions are in synchronization and that the unwanted table definitions no longer exist in either CA Datacom/DB.

1. Modify the entity-occurrence definitions in CA Datacom/DB for the database structure, deleting the unwanted tables.
2. Execute the CA Datacom/DB CATALOG function on the database.

SQL Accessible Tables

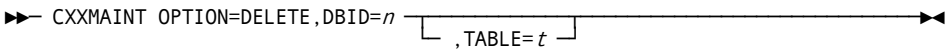
To delete tables accessed by SQL, use the SQL DROP TABLE statement (see the *CA Datacom/DB SQL User Guide*) which ensures that the Data Definition Directory (DDD) database remains synchronized with the Directory and CA Datacom/DB.

Note: A table can be deleted with the database open in the Multi-User Facility, as long as a specified table has no open User Requirements Tables.

If a table is a partition (Child), it may not be deleted. A Full Parent table may be deleted, however, with the result that all partitions (Children) are deleted.

How to Use

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=DELETE

Requests the option to delete database definitions from the Directory.

,DBID=

Specifies the ID of the database being deleted or containing the table being deleted.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Optional Keyword

,TABLE=

Identifies a table that is to be deleted.

If this parameter is omitted, the database is deleted. If any tables still exist in the database to be deleted, those tables are also deleted.

Valid Entries:

DATACOM-NAME of the table that resides in the database specified in the DBID= parameter

Default Value:

(No default)

Example JCL

The following shows the command to delete table A38 from the Directory definition for database 238.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=DELETE,DBID=238,TABLE=A38
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT DELETE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
      ❶              CXXMAINT OPTION=DELETE,DBID=238,TABLE=A38

                    FUNCTION=CXXMAINT
                       DBID=238
      ❷                OPTION=DELETE
                       TABLE=A38

      ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

      ❹          *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

Example JCL

The following shows the command to delete database 238 from the Directory.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=DELETE,DBID=238
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT DELETE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
      ❶              CXXMAINT OPTION=DELETE,DBID=238

                    FUNCTION=CXXMAINT
                       DBID=238
      ❷                OPTION=DELETE


      ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

       ❹          *                                   REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT FORMAT0102 (Base Level Full Version 12.0 Features)

Use this function to specify the ID of the database or databases to be altered from FORMAT 1 to FORMAT 2.

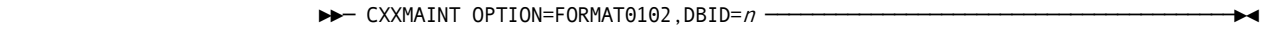
When to Use

When upgrading a Directory (CXX) from the previous release (Version 11.0, also known as r11) to Version 12.0, you are expected to run in the compatibility mode of the previous release for a few weeks and use only the Version 12.0 features that are compatible. This process ensures that, should a fallback be needed, the downgrade of the CXX can be simple and fast. This process is protected by a database level format. The use of FORMAT 1 is compatible between Versions 11.0 and 12.0. FORMAT 2 includes features supported by Version 12.0 alone. When the time for running in the compatibility mode of the previous release passes, all databases need to be set to FORMAT 2 to allow the additional Version 12.0 features not supported by the previous release. Those additional features include the following, in which Version 12.0:

- Activates key level status, where each key can be loaded or not loaded, allowing the use of the RETIX function with the MINIMAL option.
- Allows dynamic extend to be provided in cylinders or tracks, where using cylinders allows larger extent sizes.
- Allows a database index to include a Multi-Data Set Index area.

How to Use

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=FORMAT0102

Specifies to use the FORMAT0102 option.

,DBID=*n*

Specifies the ID of the database or databases to be altered from FORMAT 1 to FORMAT 2, where *n* can be 1 to 5000.

Note: It is possible to have different databases in the CXX at different formats for use in testing and by QA, but not for use in production systems.

Valid Entries:

DATACOM-ID of the database or databases

Default Value:

(No default)

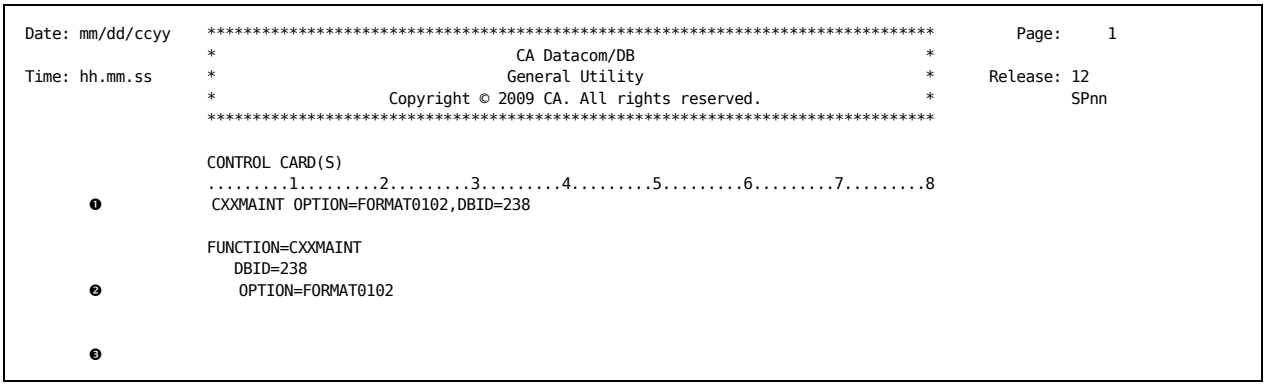
Example JCL

The following shows the command to use the OPTION=FORMAT0102 option for the database with a DBID of 238.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see .
// ASSGN SYSnnn,device-type,VOL=volser,SHR
// DLBL CXX,'cxx',,DA                              Directory data set
// EXTENT SYSnnn,volser
// EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=FORMAT0102,DBID=238
/*
/&
* $$ EOJ
```

Sample Report: CXXMAINT FORMAT0102



```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             CXXMAINT OPTION=FORMAT0102,DBID=238

                    FUNCTION=CXXMAINT
                       DBID=238
       ❷                OPTION=FORMAT0102


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

      ❹             *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT FORMAT0201 (Base Level Restrict to Previous Release Compatible Features)

Use this function to specify the ID of the database or databases to be altered from FORMAT 2 to FORMAT 1.

When to Use

User production and test systems are intended to have all databases in compatible FORMAT 1 during upgrading from the previous release (Version 11.0, also known as r11) to Version 12.0 and to remain in that format until no risk of fallback exists. When all risk of fallback has ceased to exist, all databases can be changed to full Version 12.0 feature support by using FORMAT 2. This option is therefore not intended for user-systems. It exists for special cases where individual databases need to be in a different format to allow for testing or use by QA.

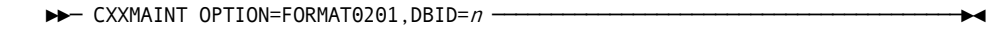
Considerations for Using FORMAT0201

Only a database in FORMAT 2 that has not used features of Version 12.0 has no issues with the change from FORMAT 2 to FORMAT 1. Issues exist for each database using Version 12.0-only features. The possible issues and how to handle them are as follows:

- If an index area or data area are currently defined as having dynamic extend selected in cylinder units, the number is divided by the tracks per cylinder and the result saved as the number of tracks. However, if the number is 64k or larger, the number is set to 64k-1. This can be an issue of concern and a reason why users are asked to not use FORMAT 2 and Version 12.0 features while there is any risk of a fallback.
- If the database has Multi-Dataset Index areas, all index areas other than the IXX have the definition removed from the Directory (CXX) and the index for the table set as not loaded.

How to Use

Multi-User mode is required to execute this command. Execute DBUTLTY with the command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=FORMAT0201

Specifies to use the FORMAT0201 option.

,DBID=*n*

Specifies the ID of the database or databases to be altered from FORMAT 2 to FORMAT 1, where *n* can be 1 to 5000.

Note: It is possible to have different databases in the CXX at different formats for use in testing and by QA, but not for use in production systems.

Valid Entries:

DATACOM-ID of the database or databases

Default Value:

(No default)

Example JCL

The following shows the command to use the OPTION=FORMAT0201 option for the database with a DBID of 238.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=FORMAT0201,DBID=238
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT FORMAT0201

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
      ❶              CXXMAINT OPTION=FORMAT0201,DBID=238

                    FUNCTION=CXXMAINT
                       DBID=238
      ❷                OPTION=FORMAT0201


      ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

       ❹          *                                    REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

CXXMAINT PURGE (Remove Database from Directory)

The PURGE option flags the Directory to indicate that the specified database does not exist. It performs none of the Directory Block Reclamation done by the DELETE option. This option removes all information about the database from the Directory (CXX).

When to Use

Do not use PURGE unless DELETE fails.

Before using PURGE, review CXXMAINT DELETE (Delete CXX Definitions) (see page 253) to ensure that the DELETE request was submitted correctly.

It may be necessary to use PURGE when the following sequence of events have occurred:

1. You deleted a database definition from CA Datacom/DB.
2. You used CXXMAINT OPTION=DELETE,DBID=n to delete that database from the Directory (CXX).
3. You received return code 13 and the DELETE option did not complete.

In this case, use the PURGE option to remove the database. Then, reclaim space by executing the BACKUP, INIT and LOAD CXX functions.

This option is provided to minimize the recovery process of the Directory in the case of hardware failure errors, CA Datacom/DB failures, or misuse of the RESET function. Without this option, these conditions can require reloading the entire Directory which requires that all users terminate processing. Current information entered since the time of the backup is lost.

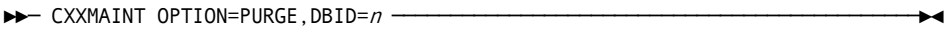
The PURGE option allows you to eliminate the problem database and restore only that database from a backup or rebuild it through CA Datacom/DB.

SQL Processing

If you have databases which are accessed by SQL, maintaining synchronization between the Directory, the DATA-DICT database, and the Data Definition Directory (DDD) database is critical for SQL operations. When you use the PURGE option, make certain you retain this synchronization.

How to Use

Multi-User mode is required to execute this command. Use the following format for the PURGE option.



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=PURGE

Requests the CXXMAINT PURGE option.

,DBID=

Identifies the database being purged.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Example JCL

The following shows the command to purge database 238.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        CXXMAINT   OPTION=PURGE,DBID=238
 /*
 /&
 * $$ EOJ
```

Sample Report: CXXMAINT PURGE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             CXXMAINT OPTION=PURGE,DBID=238

                    FUNCTION=CXXMAINT
                       DBID=238
       ❷               OPTION=PURGE


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

       ❹          *                                   REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

Chapter 9: DBTEST (Execute Test/Analysis Programs)

The DBTEST function of DBUTLTY is a debugging and analysis tool. You can use it to execute any of three testing and debugging programs:

DBTSTPR

Is the CA Datacom/DB test program. It executes data access commands without requiring you to code a test program or assemble a User Requirements Table. The output results of each command can be printed.

DBIDCPR

Is the CA Datacom/DB Index Area analysis program which allows you to review an Index Area (IXX) for correctness.

DBIDPPR

Is the CA Datacom/DB Index Area population analysis program. The DBIDPPR program prints an analysis of the distribution of key values for each level of the Index and totals for each level.

Important! The DBTEST function must be preceded by the user requirements definitions, namely DBURSTR, DBURTBL, and DBUREND. These are considered a set. The control statements containing the data access command requests are expected to follow the DBTEST function.

The DBURSTR, DBURTBL, and DBUREND functions are nearly identical to the macros required to assemble a User Requirements Table. In most cases, you can use the input to assemble these macros as input to the DBTEST process. Parameters not supported and the macros in which they occur are in the following list:

Macro	Unsupported Parameters
DBURSTR	CBSIO= CSECT= JOBSTAT= OPTMIZE= TIMEMIN= TIMESEC= WRITE=
DBURTBL	AREANAM= DBFLSUB=

Important! You cannot run any other DBUTLTY function in the same job step as DBTEST. All the input that follows the DBTEST command is considered a command or request for the DBTSTPR program.

The DBTSTPR program can run in Single User mode if NO is specified for the MULTUSE= keyword. To invoke the CA Datacom/DB test program, execute DBUTLTY using the commands described on the following pages. Maintain the command input order described below, repeating DBURTBL multiple times to build the applicable User Requirements Table entries.

DBTEST Command 1 (DBURSTR)

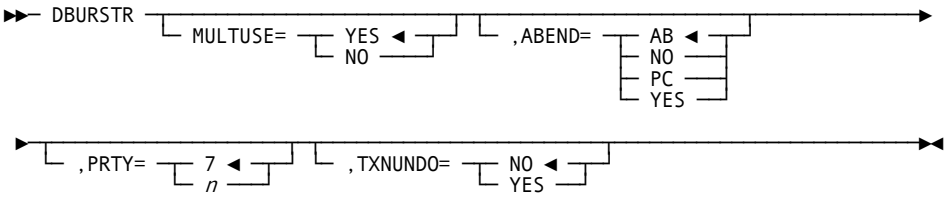
The first command, DBURSTR, specifies the start of the User Requirements Table for the CA Datacom/DB test program to use. To accept all the keyword defaults for the DBURSTR command, omit the command from the job step.

This function communicates with one Multi-User Facility. You can use it with more than one Multi-User Facility when you use the DBURSTR command and specify MULTUSE=YES. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

How to Use

To specify the start of the User Requirements Table for the CA Datacom/DB test program to use, execute the following command.



Command

DBURSTR

Specifies the start of the User Requirements Table for the CA Datacom/DB test program to use. You must specify one of the following keywords, if you include the DBURSTR command in your job step.

Optional Keywords

,ABEND=

Designates whether CA Datacom/DB or the program controls abnormal termination processing.

AB specifies that all abnormal terminations allowed by the operating system are intercepted. CA Datacom/DB handles termination processing.

PC specifies that only program check terminations are intercepted. CA Datacom/DB handles termination processing.

YES specifies the functions of both AB and PC. All abnormal terminations allowed by the operating system and program check terminations are intercepted. CA Datacom/DB handles termination processing.

NO specifies that CA Datacom/DB does not intercept abnormal processing.

Valid Entries:

AB, NO, PC, or YES

Default Value:

AB

MULTUSE=

Specifies whether the program directs requests to the Multi-User Facility.

YES causes the Multi-User interface program to be loaded and all requests to be made through the Multi-User Facility.

NO indicates the program runs in Single-User mode.

Valid Entries:

NO or YES

Default Value:

YES

,PRTY=

Specifies the priority level of the program within the Multi-User Facility.

Valid Entries:

An integer value from 1 through 15

Default Value:

7

,TXNUNDO=

Specifies whether this task runs with the Transaction Backout option. This option is ignored unless logging is being performed.

Since maintenance commands are not allowed with DBTEST, this parameter is only provided for compatibility with the DBURSTR macro. Any value you supply is ignored.

Valid Entries:

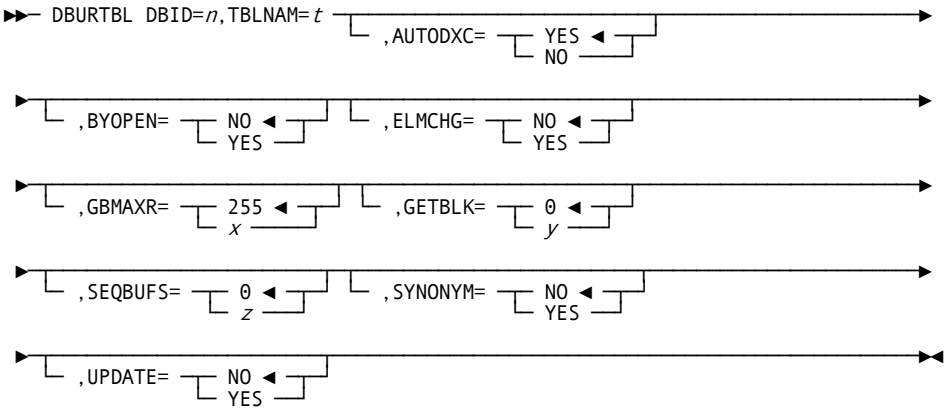
NO or YES

Default Value:

NO

DBTEST Command 2 (DBURTBL)

The second command, DBURTBL, generates an entry for a CA Datacom/DB table you are accessing in the submitted requests. Enter the DBURTBL function multiple times between DBURSTR and DBUREND to generate a User Requirements Table with access to multiple tables.



Command

DBURTBL

Generates an entry for a CA Datacom/DB table you are accessing in the submitted requests.

Required Keywords

DBID=

Specifies the identifier of the database being accessed.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,TBLNAM=

Specifies the DATACOM-NAME of a table in the database specified in the DBID= parameter.

Valid Entries:

DATACOM-NAME of a table in the database

Default Value:

(No default)

Optional Keywords

,AUTODXC=

Specifies whether GETIT, GSETL, GETPS, RDUNX, RDUNK, RDUBR, and RDUNE commands automatically drop exclusive control established by any previous command from the same request area.

Valid Entries:

NO or YES

Default Value:

YES

,BYPOPEN=

Specifies whether the table defined in this function is to be bypassed during open and close processing.

Valid Entries:

NO or YES

Default Value:

NO

,ELMCHG=

Specifies whether the element list for sequential operations can be changed between GETIT or GETPS commands. Specifying NO improves execution time because verification of the element list is not required between commands. If you assign a value other than zero to GETBLK=, the value for ELMCHG= must be NO.

Valid Entries:

NO or YES

Default Value:

NO

,GBMAXR=

Specifies the maximum number of entries allowed in the blocked GETIT buffer. If UPDATE=YES is specified, the value of this parameter cannot exceed the value specified for the Multi-User startup option EXCTLNO (see the *CA Datacom/DB Database and System Administration Guide*).

Valid Entries:

An integer value from 2 through 255

Default Value:

255

,GETBLK=

(Multi-User only.) Specifies the buffer size used for blocking GETIT or GETPS commands. The buffer must be large enough to contain at least two request areas and two work areas. If you assign a value other than zero to GETBLK=, the value for ELMCHG= must be NO, and the value for AUTODXC= must be YES.

Valid Entries:

The integer value 0, or an integer value from 4096 through 61440

Default Value:

0

,SEQBUFS=

Specifies the number of data buffers the program can use for single block or multiple block read-ahead sequential processing. Use only positive numbers.

Valid Entries:

An integer value of 0 or from 2 through 128

Default Value:

0

,SYNONYM=

Specifies whether other tables with the same name are included in other DBURTBL entries in this User Requirements Table generation. If you enter YES, all requests made for this table must include the database identifier in the request area.

Valid Entries:

NO or YES

Default Value:

NO

,UPDATE=

Specifies update authorization. Although you cannot use maintenance commands with the test utility, UPDATE=YES is required if you intend to test commands that acquire exclusive control.

Valid Entries:

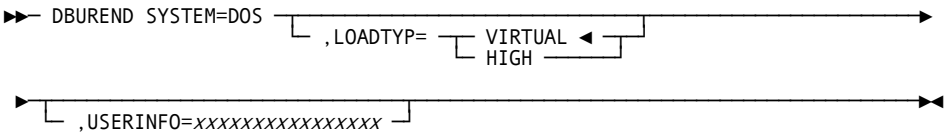
NO or YES

Default Value:

NO

DBTEST Command 3 (DBUREND)

The third command, DBUREND, indicates the end of the User Requirements Table.



Command

DBUREND

Indicates the end of the User Requirements Table.

Required Keyword

SYSTEM=

Defines the operating system. For z/VSE, specify DOS.

Optional Keywords

LOADTYP=

Specifies where the CA Datacom/DB code is loaded.

Valid Entries:

VIRTUAL or HIGH

Default Value:

VIRTUAL

,USRINFO=

Specifies one to sixteen characters of user information to be placed as a marker in this User Requirements Table.

Valid Entries:

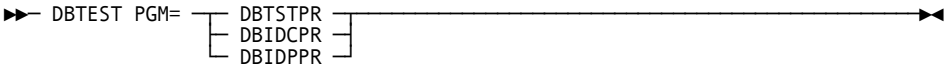
A 1- to 16-character string

Default Value:

(No default)

DBTEST Command 4 (DBTEST)

The fourth command, DBTEST, identifies the program to be executed.



Command

DBTEST

Invokes the function to execute a test or analysis program.

Required Keyword

PGM=

Specifies the name of the program to execute.

Valid Entries:

DBTSTPR (see description on DBTEST DBTSTPR (CA Datacom/DB Test Program) (see page 277))
DBIDCPR (see description on DBTEST DBIDCPR (Index Analysis Program) (see page 283))
DBIDPPR (see description on DBTEST DBIDPPR (Index Population Analysis Program) (see page 286))

Default Value:

(No default)

DBTEST DBTSTPR (CA Datacom/DB Test Program)

When to Use

Use the DBTSTPR program to test a sequence of native CA Datacom/DB data access commands. You can examine return codes and data returned in the work area.

You cannot use DBTSTPR to test:

- Maintenance commands (UPDAT, DELET, ADDIT)
- Logging commands (LOGxx)
- Compound Boolean Selection commands (SELxx)
- Commands that require two key values, for example: CNTKR
- SQL statements (if you have the SQL Mode)

Note: The Multi-User Facility does *not* recognize DBTSTPR as a DBUTLTY controlled job with special priviledges.

Control Statements

When you specify DBTSTPR, you must supply control statements immediately following the DBTEST statement in your job stream. Two types of control statements are accepted by the CA Datacom/DB test program. The first type specifies the DUMP printing options. These options remain in effect until the next DUMP statement. The second type of control statement is the command statement. You can use any valid CA Datacom/DB random read, sequential read, or locate command on this statement. The format of each of the control statements follows.

Print Control Statement

The Print Control statement is formatted as follows:

Starting Column	Value and Description
1	DUMP= Required.
6	Enter one of the following values. **P** Print input statements and one line of returned elements in character format. The output results of the P option are directed to SYSLST. **R** Print request area and work area in dump format. **B** Print a memory dump of the Master List, including buffers. **L** Print a memory dump of the Master List, excluding buffers. **M** Use at the direction of Technical Support. Print an internal trace of buffer requests (hex). **N** Print nothing and reset (turn off) all print options.

You can enter multiple print control statements. The effect of each control value is cummulative until you enter a print control statement with DUMP=N specified.

Command Control Statement

The Command Control statement is formatted as follows:

Starting Column	Field Length	Description
1	5	Requested command code.
6	3	Requested table name.
9	5	Requested key name.
14	1	Null - not used.

Starting Column	Field Length	Description
15	1	Special control. Can be blank, or can indicate the number of elements entered (1, 2, 3). The Element List start column varies as shown by the table below. The use of a number (1, 2, 3) is to allow the key value length to increase as needed. The maximum length is limited to 33 positions.
16	4	Optional database ID.
20	15	Key value for commands that require a value. See the *CA Datacom/DB Programming" Guide* for individual command requirements.
35	24	Element List. Varies with the number used in column 15.
70	2	Repeat factor to cause this command to be executed from 01 through 99 times.

Any statement with an asterisk (*) in column 1 is treated as a comment statement and printed if the P option is on; otherwise, it is ignored.

To use a key value larger than 15 bytes in the Command Control Statement, modify the start column of the element list by placing a control character in column 15 as follows:

Column 15 Contents	Maximum Key Length	Element List Starting Column	Maximum Element List Entries
Blank	15	35	4
3	21	41	3
2	27	47	2
1	33	53	1

Example DBTSTPR JCL

The following shows the commands to build a User Requirements Table for the PMF table in database 1, allowing no update ability. The DBTEST command invokes the CA Datacom/DB test program to locate the first key value with a state/zip code key of TX75000 or greater.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // DLBL PMF001,'db001.pmf',,DA                     Data area PMF in DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          DBURSTR  MULTUSE=YES
          DBURTBL  DBID=1,TBLNAM=PMF
          DBUREND  SYSTEM=DOS
          DBTEST   PGM=DBTSTPR
 DUMP=P
 *CMD TBLKEYNM  DBI KEY            ELEM*4
 LOCKGPMFSTZIP      TX75000
 /*
 /&
 * $$ EOJ
```

Sample Report: DBTSTPR

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBURSTR  MULTUSE=YES

                    FUNCTION=DBURSTR
       ❷                MULTUSE=YES


       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBURTBL  DBID=1,TBLNAM=PMF

                    FUNCTION=DBURTBL
       ❷                DBID=1
                        TBLNAM=PMF


       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBUREND SYSTEM=DOS

                    FUNCTION=DBUREND
       ❷                SYSTEM=DOS


       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     4
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBTEST   PGM=DBTSTPR

                    FUNCTION=DBTEST
       ❷               PGM=DBTSTPR



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     5
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************
 ❹
DUMP=P                                                                                 5/25/2003       12.01.50
*CMD TBLKEYNM  DBI KEY           ELEM*4   ❺                                                          12.01.50
LOCKGPMFSTZIP      TX75000   ❻                                                                      12.01.50
TX75243                                                                                 LOCKGPMFSTZIP  12.01.50
  ❼                                                                                       ❽             ❾
```

❹

The dump option statement entered requesting input statement display.

❺

One line of returned elements.

❻

The first command - a Locate Key.

❼

Key value returned from successful execution of the LOCATE KEY.

❽

The request area followed by a return code.

❾

The time at which each request was executed.

DBTEST DBIDCPR (Index Analysis Program)

This special utility is a support tool that allows you to review an Index Area (IXX) for correctness.

It scans the Index Area (IXX) for a single database. If the Index Area (IXX) has errors, the utility snaps/dumps the Master List and terminates. If the utility finds no error, it terminates without a report, but with a good exit.

The DBIDCPR utility only scans the Index Area (IXX). It does not scan the data area, so no verification is done that the Index matches the data.

When to Use

Use DBIDCPR only at the direction of Technical Support.

Incomplete Indexes

The utility accepts and skips over an incomplete Index when deleted blocks are found in the Index chains or when a block is not indexed. These situations are temporary and do not indicate an incorrect Index. However, they could indicate a problem, so an option is provided to snap/dump a Master List each time either situation is encountered. This program is selected by setting the value for the PRTY= parameter greater than 7 in the DBURSTR command.

Not Available with External Security

You cannot run this program if the Directory (CXX) is secured with external security.

How to Use

To execute this program:

1. Code a DBURSTR macro with MULTUSE=NO. (See DBTEST Command 1 (DBURSTR) (see page 270).)
2. Code a single DBURTBL macro specifying the database to be processed and a table in that database. The table must be able to be opened normally. (See DBTEST Command 2 (DBURTBL) (see page 272).)
3. Code a DBUREND macro specifying the operating system in the SYSTEM= parameter. (See DBTEST Command 3 (DBUREND) (see page 275).)
4. Code the DBTEST function specifying the program (PGM=) as DBIDCPR. (See DBTEST Command 4 (DBTEST) (see page 276).)

Example DBIDCPR JCL

An Index Analysis is being performed for table PNF in database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA               Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // DLBL PMF001,'db001.pmf',,DA                 Data area PMF in DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          DBURSTR  MULTUSE=NO
          DBURTBL  DBID=1,TBLNAM=PMF
          DBUREND  SYSTEM=DOS
          DBTEST   PGM=DBIDCPR
 /*
 /&
 * $$ EOJ
```

Sample Report: DBIDCPR

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBURSTR  MULTUSE=NO

                    FUNCTION=DBURSTR
       ❷               MULTUSE=NO



       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBURTBL  DBID=1,TBLNAM=PMF

                    FUNCTION=DBURTBL
       ❷               DBID=1
                       TBLNAM=PMF



       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBUREND  SYSTEM=DOS

                    FUNCTION=DBUREND
       ❷               SYSTEM=DOS



       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     4
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                     *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBTEST   PGM=DBIDCPR

                    FUNCTION=DBTEST
       ❷               PGM=DBIDCPR



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

After DBIDCPR scans the Index Area (IXX) for a single database, it terminates, if it finds no errors, with a good exit but without generating a report. If errors are found, however, DBIDCPR snaps/dumps the Master List (see Master List Dump Example (see page 175)) and terminates.

DBTEST DBIDPPR (Index Population Analysis Program)

The DBIDPPR program prints an analysis of the distribution of key values for each level of the Index and totals for each level.

When to Use

Use DBIDPPR only at the direction of Technical Support in response to a problem because the output produced can be quite large.

DBIDPPR Output

For each low-level (0) DXX block a line is printed with the total number of record pointers, the number of key value data changes, the key ID, and the length of the stored portion of the key data followed by the highest key value in that block. However, only character type data is displayed in the report.

The low value is either binary zeros for the first entry or for the high value of the previous entry. See the DBIDPPR Report on the following pages.

The high value for the last entry of a key ID is binary ones. At the end of every high-level block the totals for that block and lower blocks are printed.

How to Use

To execute this program:

1. Code a DBURSTR macro with MULTUSE=NO. If MULTUSE=YES is coded, it is treated as MULTUSE=NO. (See DBTEST Command 1 (DBURSTR) (see page 270).)
2. Code a single DBURTBL macro specifying the database to be processed and a table in that database. The table must be able to be opened normally. (See DBTEST Command 2 (DBURTBL) (see page 272).)
3. Code a DBUREND macro specifying the operating system in the SYSTEM= parameter. (See DBTEST Command 3 (DBUREND) (see page 275).)
4. Code the DBTEST function specifying the program (PGM=) as DBIDPPR. (See DBTEST Command 4 (DBTEST) (see page 276).)

Example DBIDPPR JCL

This example produces an Index Population Analysis for table PMF in database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // DLBL PMF001,'db001.pmf',,DA                     Data area PMF in DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          DBURSTR  MULTUSE=NO
          DBURTBL  DBID=1,TBLNAM=PMF
          DBUREND  SYSTEM=DOS
          DBTEST   PGM=DBIDPPR
 /*
 /&
 * $$ EOJ
```

Sample Report: DBIDPPR

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBURSTR  MULTUSE=NO

                    FUNCTION=DBURSTR
       ❷               MULTUSE=NO



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBURTBL  DBID=1,TBLNAM=PMF

                    FUNCTION=DBURTBL
       ❷               DBID=1
                       TBLNAM=PMF


       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBUREND  SYSTEM=DOS

                    FUNCTION=DBUREND
       ❷               SYSTEM=DOS



       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     4
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                     *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DBTEST   PGM=DBIDPPR

                    FUNCTION=DBTEST
       ❷               PGM=DBIDPPR



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

Portions of selected pages of the report appear below.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     5
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


    ❶         ❷           ❸       ❹     ❺       ❻

LEVEL  TOTAL PTR'S  DATA CHANGES  KID  LENGTH  HIGH VALUE

    0          368           140    1       4
    0          160            57    1       4
    0          382           177    1       4
    0          419           290    1       4
    0          292           245    1       4
    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .

    1        3,323         1,611        ❼

    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .
    .            .             .    .       .
    0           78            27  203      66  SYSADM                    SYSTABAUTH             TA
    0          158            59  203      65  SYSADM                    TEXT                   H

    1        3,943         1,409

    2      112,059        42,563        ❽
```

❶

Index level.

❷

Total number of record pointers.

❸

Number of key value data changes.

❹

Key ID.

❺

Length of the stored portion of the key data.

❻

Highest key value in this block. (Only character type data is displayed.)

❼

Totals for the preceding block.

❽

Totals for the preceding block and grand totals.

Chapter 10: DEFRAG (Defragment Index)

The DBUTLTY DEFRAG function defragments or compresses the DXX, that is, the "lowest" level of the IXX that contains key values and pointers to actual rows. The DBUTLTY DEFRAG function provides a simpler way to accomplish much the same results as performing an INIT to the Index Area (IXX) then doing a RETIX or LOAD of the DBID.

In order to INIT the IXX, the DBID must be completely offline to other users. The DEFRAG function runs through the Multi-User Facility and allows the DBID to remain online and usable to other applications during the DEFRAG.

Overview

CA Datacom/DB manages index space usage at the block level. The process of adding index entries and pointers adds to an existing index block until it is full. At that point the index must be split. CA Datacom/DB chooses intelligent split points based on the pattern of maintenance to the existing block and, from there, continues filling existing blocks until they must split.

Based on the above add patterns, each low level (DXX) block represents a given key range into which future adds may fall. If only some of the pointers or entries are deleted in a given block, the space formerly occupied by them is available for new adds, but only within the key range of that block. When future adds are all outside this range, index fragmentation results.

If all entries are deleted from an index block, it is reclaimed as a free block and is then available for future use. In this case, no fragmentation exists.

Instead of defragmenting the IXX by performing an INIT of the IXX then doing a RETIX or LOAD of the DBID, you can simply use the DEFRAG function. The DEFRAG function does not alter the high level IXX blocks or lower the number of index levels.

DEFRAG SMP locking is used to interact with backwards processing in the index, such as when a LOCBR command is issued. Information about DEFRAG SMP locking can be found in the SMP Informational Report obtained by specifying it with the SMPTASK Multi-User startup option. See the listing of the DEFRAG event in the table shown in the section SMP Informational Report.

Processing

The DEFRAG function is necessarily an I/O instensive process. Before two blocks can be merged, they are written to DASD, if they are already in the pipeline. Each write in the merge process is an immediate write.

When two blocks are chosen to be merged, their block numbers and information are written to IXX control block. If the Multi-User Facility fails, restart (or any open of the DBID for update) uses this information to ensure that the index is consistent in regard to those two blocks. If you choose not to restart the DBID and then attempt to open the DBID as read-only, the open fails with a return code 43(005).

The merged-to block is always written first. The merged-from block is then deleted and written. Write I/O errors during the DEFRAG process are unlikely. If any should occur, however, they are handled in the following way. If it is on the first block (the merged-to block) the block is simply marked invalid, which requires a future user of the block to reread it from DASD, and the DEFRAG terminates. If it is on the second block (the newly deleted block), the index is marked in the Multi-User Facility memory as unusable. No I/O reads at the DXX level are allowed. You should close the DBID and correct the cause of the I/O error. When the DBID is reopened for update, the same correcting code that exists for system failure runs and deletes the second block. If the I/O error persists, the index has to be initialized and rebuilt using the RETIX function, that is, dealt with the same as any other I/O error problem.

Restrictions

The DEFRAG function cannot be used with entry points other than DBNTRY, that is, cannot be used with *old* entry points. If the IXX is already open in the MUF when DEFRAG begins, the DEFRAG process checks to see if there have been any old entry point users for that DBID since the physical IXX was opened. If the answer to that test is no, DEFRAG continues. If the answer to that test is yes, DEFRAG ends with a return code 94(072).

While DEFRAG is running, you can run an old entry point request against the DEFRAG DBID, but if that is done the DEFRAG function of DBUTLTY stops with a return code of 94(072).

Stopping DEFRAG

The index defragmentation is subject to REQABORT if you wish to stop it while it is running. This REQABORT is non-destructive and leaves the index perfectly usable.

Security Considerations

If DBUTLTY is externally secured, you must have access to this DTUTIL resource: cxxname.DBUTLTY.DEFRAG. There are no underlying table rights.

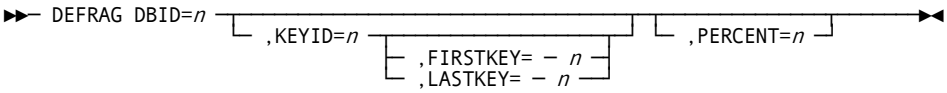
DEFRAG

When to Use

Use the DEFRAG function when you want to defragment the index (IXX).

How to Use

Multi-User mode is required to execute this command. Execute the DEFRAG function using the following command format:



Command

DEFRAG

Invokes the DEFRAG function.

Required Keywords

DBID=

Specify the name of the DBID that you want to use.

Valid Entries:

a valid DBID

Default Value:

(No default)

Optional Keywords

,KEYID=*n*

Specify the key ID that you want to use.

Note: A normal DEFRAG does *not* include this keyword. KEYID= is provided for *special* cases where past executions have been analyzed and results have shown that one KEYID (or more) does not derive benefits from the DEFRAG.

Valid Entries:

a valid key ID

Default Value:

(No default)

,FIRSTKEY= and/or ,LASTKEY=

If you specify KEYID=, you can use FIRSTKEY= and/or LASTKEY= to limit DEFRAG to a segment of the specified key. We recommend you allow FIRSTKEY=/LASTKEY= to default. The DEFRAG function normally defaults to include the full specified key range from low values to high values. FIRSTKEY= overrides the default starting position while LASTKEY= overrides the default ending position, limiting what is processed by DEFRAG. If one keyword is specified and the other is not specified, a default value is selected for the missing keyword. Normal CA Datacom key lengths are from 1 through 180 bytes. The key value data entered for FIRSTKEY= and LASTKEY= can be from 0 (zero) through 59 bytes long. You only need to code the number of positions of the key value that are significant to you, but be aware that the value coded is *left justified regardless of key field data type.* The utility pads the low order or remaining positions of the key value with low values on FIRSTKEY= and high values on LASTKEY=. To DEFRAG the entire key in one step, specify neither FIRSTKEY= nor LASTKEY=. Alternately, you could specify FIRSTKEY=00 and/or LASTKEY=00 to cause processing to start at low values and end at high values. The length of zero indicates that the padded key values (low values and high values) are to be used.

The data is not edited or interpreted. Therefore, if the key value data you need to enter is in binary, you must set up the JCL using a hexadecimal mode display and enter the key value data in hexadecimal. This data can contain blanks, commas, and any other special characters. CA Datacom/DB cannot edit this data for syntax, so the control statement can *look* invalid, especially when it includes blanks or commas or key value data containing English words. For example, FIRSTKEY=12121212121212 means the length is 12, key value is 121212121212. Another example, FIRSTKEY=10FIRSTKEY10 means the length is 10, key value is FIRSTKEY10.

Valid Entries:

A 2-byte length (in a range of 00 through 59) followed by a character string of key value bytes (for the length specified) that represents the key value.

Default Value:

The starting/ending position of the full key range from low to high values.

,PERCENT=*n*

PERCENT= defaults to 100, which means all DXX blocks that can be merged will be merged, that is, any block that is less than completely full is a merge candidate. PERCENT=0 (zero) is treated as PERCENT=100. Coding a value between 1 and 99 means you want to tune which DXX blocks qualify as a merge candidates. Each DXX block has its current populated percentage compared to the number specified. If it is less than the specified percent, it is a merge candidate.

Note: We recommend that you do not code a value for PERCENT= but allow it to default to 100.

Valid Entries:

0 thru 100, where specifying a zero is the same as specifying 100

Default Value:

100

Example JCL

The following shows the command to defragment an index with DBID=781.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN  SYSnnn,DISK,VOL=volser,SHR
 // DLBL   IXX781,'ixx.db781',,DA
 // EXTENT SYSnnn,volser
 // DLBL   781A01,'A01.db781',,DA
 // EXTENT SYSnnn,volser
 // EXEC   DBUTLTY,SIZE=768K
    DEFRAG DBID=781
 /*
 /&
 * $$ EOJ
```

Sample Report: DEFRAG

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             DEFRAG  DBID=781

                    FUNCTION=DEFRAG
       ❷               DBID=00781


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
Base:   781         ******************************************************************************** Directory: cxxname

                                                DEFRAG REPORT
               BLOCKS         BLOCKS         BLOCKS         BLOCKS         PERCENT
KEY ID         BEFORE       COMBINED        DELETED          AFTER        DELETED  NOTES

     1             65             23             23             42           35.38
     2            240             26            174             66           72.50
     3              0              0              0              0            0.00
     4              0              0              0              0            0.00
     8             45             29             29             16           64.44
    11             10              0              0             10            0.00
    12             16              0              0             16            0.00
 39321              1              0              0              1            0.00  INDEX FREE SPACE
 39327              1              0              0              1            0.00  DATA FREE SPACE
TOTALS            378             78            226            152           59.78
```

Note: In the report all user keyids which are processed have statistics printed. Any special keyid (those with notes, that is, INDEX FREE SPACE and DATA FREE SPACE) only print statistics if it has at least one DXX block. This example does not contain an exhaustive list of the special keyids.

Also, note that there is a printed statistic of "blocks deleted". This contains two things. The first is DXX blocks which DEFRAG found already deleted but which the index queue has not yet processed. The second are blocks which the DEFRAG itself deletes. These are typically blocks which are combined with other blocks, but they may also be something like a cleanup of logically deleted entries which were still physically present in the DXX when DEFRAG ran. Hence DEFRAG can be used as a form of DXX TSN cleanup.

Chapter 11: EDIT (Validate Syntax)

The EDIT function of DBUTLTY checks the syntax of one or more function keyword sets without executing the functions. When the EDIT function is specified, DBUTLTY edits all of the commands (and the keywords that follow them) for missing or invalid input. DBUTLTY displays all of the input that is accepted.

This function does not attempt to verify that any database, area, or table specified in the input actually exists. The editing is for most conditions, but not every edit occurs when the function is completely processed.

The EDIT function does not open any CA Datacom/DB tables.

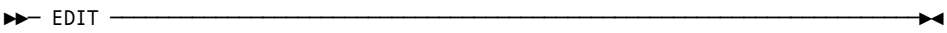
EDIT

When to Use

Use this function when setting up JCL for a DBUTLTY function for the first time or when making changes to JCL.

How to Use

You can execute this command in either Single User or Multi-User mode. To edit a stream of DBUTLTY functions, execute DBUTLTY using the following command format preceding the input stream to be edited:



Command

EDIT

Invokes the EDIT function for the input stream following this command. This must be the first input control statement in the input stream.

Example JCL

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // DLBL PMF001,'db001.pmf',,DA                     Data area PMF in DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          EDIT
          DBURSTR  ABEND=YES,MULTUSE=YES
 /*
 /&
 * $$ EOJ
```

Sample Report: EDIT

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶            EDIT

       ❷            FUNCTION=EDIT



       ❸

DB10016I - EDIT ONLY, NO FUNCTION(S) PROCESSED   ❹
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                     *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    ........1.........2.........3.........4.........5.........6.........7.........8
       ❺             DBURSTR ABEND=YES,MULTUSE=YES

                    FUNCTION=DBURSTR
       ❻               ABEND=YES
                       MULTUSE=YES


DB10016I - EDIT ONLY, NO FUNCTION(S) PROCESSED
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

❹

Message DB10016I indicates an EDIT was in process.

❺

The command that CA Datacom/DB edited, exactly as it was entered following the EDIT command.

❻

An analysis of the keywords encountered and expected with the command being edited. CA Datacom/DB flags any errors with a note in the left margin.

Chapter 12: EXTBKUP (Extract from Backup File)

The EXTBKUP function of DBUTLTY reads a backup file and writes extract of backup format data to the output data set(s).

Important! Prior to the execution of the EXTBKUP function, one or more FILEOUT functions *must* have been specified. FILEOUT functions define which tables are to be extracted by the EXTBKUP function. See FILEOUT (Define Output Data Sets) (see page 341).

A backup file contains records for one or more tables, based upon the area or database content, and optional control statements provided to the backup utility. The EXTBKUP function can extract one or more tables from a backup file, as specified by the user. The function can also expand compressed data on the backup to a new backup file with no compressed records.

EXTBKUP with FORMAT=EXTRACT outputs data in expanded form. An extract file is not compressed and contains no CA Datacom control information, but backup files always contain control information and may or may not contain compressed data. Compressed data, if present, can be in the CA Datacom form or in a user compression form, such as the CA Datacom Presspack. EXTBKUP uses the control information on the backup file to know what tables exist, the row size, and compression options.

EXTBKUP

The FILEOUT function (see FILEOUT (Define Output Data Sets) (see page 341)) is used to define one or more output data sets for EXTBKUP. If the table is empty, FILEOUT creates a valid null output data set.

EXTBKUP usually writes each data record as a fixed-length record in a fixed-block format.

Output files have a block size as specified in the JCL or a default of the operating system maximum, based upon device type. TAPE is blocked up to 32767, and DISK is blocked up to 27998.

When using the EXTBKUP function, an error occurs after one the following scenarios:

- no FILEOUT statements were provided, or if
- The input file is not a valid CA Datacom backup file produced by r10 or later, that is, backups produced by CA Datacom releases before r10 are not supported.

EXTBKUP reads the control records at the start of the backup and edits them to see if the backup represents all tables selected by the FILEOUT statements.

Note: Tables are not required to have rows present.

Prior to the EXTBKUP function, one or more FILEOUT functions must have been specified to define which tables are to be extracted.

When to Use

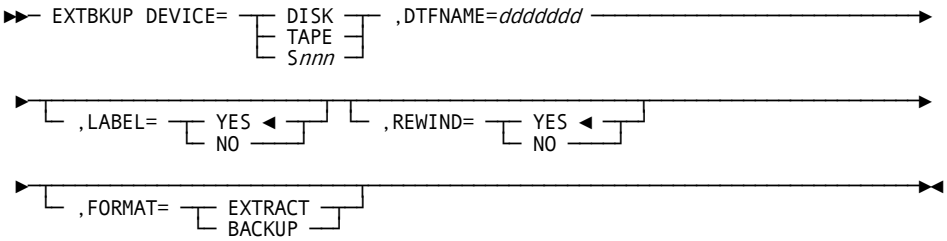
Use the EXTBKUP function when you intend to process table information for use with the CA Datacom/DB Record Migration Facility utility (DDRMFLM) in fixed-block format or for use outside the CA Datacom environment.

You can also use EXTBKUP FORMAT=BACKUP to create an uncompressed copy of of the backup file.

How to Use

Multi-User mode is required to execute this command. Execute the EXTBKUP function using the following command format:

Extract from Backup File (z/VSE)



Command

EXTBKUP

Invokes the function to extract data from a backup file in expanded form.

Required Keywords

Specifies the name of the input backup file.

Valid Entries:

Any name valid to z/OS JCL specifications

Default Value:

(No default)

DEVICE=

Specifies the z/VSE device type for the input file. When you specify DEVICE=TAPE, the system logical unit (SYS) number must be 5, (SYS005). DEVICE=S*nnn* allows you to use the *nnn* to specify a different SYS number (for tape only).

Valid Entries:

DISK, TAPE, or S*nnn*

Default Value:

(No default)

,DTFNAME=

Specifies the name of the input backup file.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

Optional Keywords

,LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), specifies whether tape labels are to be processed. We recommend that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,REWIND=

Specifies action at OPEN and CLOSE. If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

FORMAT=

If you do not specify FORMAT= (or if you specify FORMAT=EXTRACT), output is one or more extracted files, as was the case prior to Version 12.0. If you specify FORMAT=BACKUP, the output is a single file in backup format with all records compressed by any option to be expanded. When FORMAT=BACKUP, you are required to have a previous single FILEOUT function to define the single output file, and in that FILEOUT function the keyword TABLE= must be set to *** (three asterisks) to specify that all tables be written to the output file.

In support of FORMAT=, message DB13279E could be received if there is an edit error (for details about the message, see the *CA Datacom/DB Message Reference Guide*).

Note: VAR or NONE are not valid entries for FORMAT= in EXTBKUP and, if specified, are considered an error.

Valid Entries:

EXTRACT or BACKUP

Default Value:

EXTRACT

Example JCL

The following shows the command to extract the data table POL in area POL in database 1 to the output data set, POLOUT. The FILEOUT statement is shown immediately before the EXTBKUP line.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL INFILE,'db001.pol',,DA                 Input file
 // EXTENT SYSnnn,volser
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL POLOUT,'polout'                        Output data
 // EXEC DBUTLTY,SIZE=750K
          FILEOUT  DTFNAME=POLOUT,TABLE=POL,DEVICE=S005
          EXTBKUP  DEVICE=DISK,DTFNAME=INFILE
 /*
 /&
 * $$ EOJ
```

The following shows the command(s) to create an extract from the backup table POL in physical sequence to POLOUT on disk. The FILEOUT statement is shown immediately before the EXTBKUP line.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL INFILE,'db001.pay',,DA                 Input file
 // EXTENT SYSnnn,volser
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
```

```
// DLBL POLOUT,'out',1,SD                          Output data
// EXTENT,volser,1,0,rel trk/blk,number trks/blks
// EXEC DBUTLTY,SIZE=750K
         FILEOUT  DTFNAME=POLOUT,TABLE=POL,DEVICE=DISK
         EXTBKUP  DEVICE=DISK,DTFNAME=INFILE
/*
/&
* $$ EOJ
```

Sample Report: EXTBKUP

Information is reported about the AREA, TABLE, TABLEID, and number of records extracted. Zero records indicated that a null extract file was created because there were no records on the backup file. Each table specified in a FILEOUT statement (see FILEOUT (Define Output Data Sets) (see page 341)) must be found in the control records for the input backup file. If any of the specified tables are not found in the control records for the input backup file, a message in the report indicates the invalid table name, and the utility terminates as a failure.

Following is a sample EXTBKUP report.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             FILEOUT DTFNAME=POLOUT,TABLE=POL,DEVICE=S005

                    FUNCTION=FILEOUT
       ❷               DTFNAME=POLOUT
                       TABLE=POL
                       DEVICE=S005

       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             EXTBKUP DEVICE=DISK,DTFNAME=INFILE

                    FUNCTION=EXTBKUP
       ❷               DEVICE=DISK
                       DTFNAME=INFILE

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                   Copyright © 2009 CA. All rights reserved.                  *               SPnn
                    ******************************************************************************** Directory: PRODCXX

BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA
DBUTLTY    42,395  mm/dd/ccyy hh.mm.ss  PRODCXX       1  NAT   NO
OUTPUT TABLE INFORMATION
AREA  TABLE  ID     RECORDS
POL   POL     4           4         ❹
```

❹

Confirmation that records were extracted from the requested table, as specified by the FILEOUT statement.

The following is a sample z/VSE EXTBKUP report.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     4
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                   Copyright © 2009 CA. All rights reserved.                  *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             FILEOUT DTFNAME=OUT1,TABLE=PAY,DEVICE=DISK

                    FUNCTION=FILEOUT
       ❷               DEVICE=DISK
                      DTFNAME=OUT1
                      TABLE=PAY

       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     5
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                     *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶            FILEOUT DTFNAME=OUT2,TABLE=PMF,DEVICE=DISK

                    FUNCTION=FILEOUT
        ❷              DEVICE=DISK
                      DTFNAME=OUT2
                      TABLE=PMF

        ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     6
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                     *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶            FILEOUT DTFNAME=OUT3,TABLE=POL,DEVICE=DISK

                    FUNCTION=FILEOUT
        ❷              DEVICE=DISK
                      DTFNAME=OUT3
                      TABLE=POL

        ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     7
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                     *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶            EXTBKUP DTFNAME=PAYBKP,DEVICE=TAPE

                    FUNCTION=EXTBKUP
        ❷              DEVICE=TAPE
                      DTFNAME=PAYBKP

        ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     8
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA
DBUTLTY    42,395  mm/dd/ccyy hh.mm.ss  PRODCXX       1  NAT   NO
OUTPUT TABLE INFORMATION
AREA  TABLE  ID     RECORDS
PAY   PAY     1         200         ❹
PMF   PMF     2         200
DEM   POL     4           4
```

❹

Confirmation that records were extracted from the requested table.

Chapter 13: EXTEND (Extend Data or Index Areas)

The EXTEND function of DBUTLTY expands an existing data area or index area and formats the space acquired. The function also updates the data or index area control record, informing CA Datacom/DB that additional space is available.

Note: You cannot use EXTEND to decrease the size of an area.

Once the EXTEND function completes, the new area is immediately available for use without any further changes to the system. Any job that starts after the completion of the EXTEND function can use this newly allocated space.

Note: Data and index areas can be dynamically extended when they become full. See the chapters on creating and maintaining data and index areas in the *CA Datacom/DB Database and System Administration Guide*.

EXTEND

When to Use

Use the EXTEND function to increase the space available in a data or index area. You can use EXTEND:

- When planning for future growth.
- When an area becomes full or nearly full.

Note: A Multi-Dataset Index area (I00-99) can only be extended if the IXX Index Area is initialized and able to be open for processing. This requirement does not exist for the IXX Index Area and data areas.

How Space Is Allocated for an EXTEND

The additional space into which a database data area or index area is to be extended can be allocated in one of three ways:

1. You can increase the number of tracks in the last extent statement of the current extents, if space exists immediately after the current last extent.
2. You can add one or more additional extent statements after the current extent statements. New extents can never be added to a volume with current extents, except the last volume.

3. You can move the data or index area to another larger data set on another volume of the same device type. This move should be done as a backup/restore process. Non-CA Datacom/DB programs exist to accomplish this move also.

   The EXTEND function compares, in this case, information from the current extents to the control record information. EXTEND then formats any new blocks based on the extent information from the IBM OPEN. The function updates the control record to reflect the new block count.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any Multi-User Facility for read.
- The database may *not* have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO*

- The database will be opened for update.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when MULTUSE=YES*

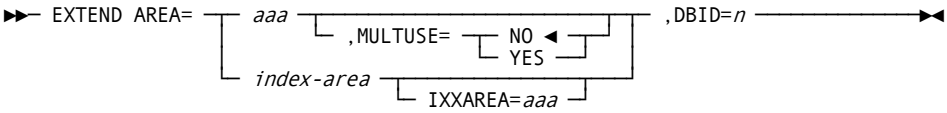
- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The area may have no table in an open URT.
- The area may *not* have any other MULTUSE=YES utility executing.
- The area must be closed.

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets EXTEND executing this Multi-User Facility this database or area.

How to Use

Multi-User mode is required to execute this command. Execute the EXTEND function using the following command format:



Command

EXTEND

Invokes the function for Index or data area expansion and formatting.

Required Keywords

AREA=

Identifies the area to be extended, where *aaa* is the name of the area to extend, and where *index-area* indicates that the index area of the database is to be extended.

Valid Entries:

the index area of the database or the DATACOM-NAME of the data area in the database

Default Value:

(No default)

,DBID=

Identifies the database for which the area expansion is required.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Optional Keyword

,MULTUSE=

(For area level DBUTLTY control only.) In support of area level DBUTLTY control, specifying MULTUSE=YES in the EXTEND function for a data area allows a single area in a database to be extended with no disabling of the other areas in the database. When MULTUSE=YES, the function executes in a format that is a combination of Multi-User Facility and Single User modes. The database involved must be able to be opened for update in the Multi-User Facility address space. The area, however, must have no tables open. Once begun, the function does not allow other tasks to open any tables in the area being processed. Other tables in the database, but not in the area, may be open for read or update in the same Multi-User Facility. The actual data set open is performed in the utility region, and the JCL requirements are applied to the utility region.

When you specify or allow the default of MULTUSE=NO in the EXTEND function for data areas, the function executes completely in Single User mode.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

NO or YES

Default Value:

NO

IXXAREA=*aaa*

The IXXAREA= keyword is optional if the database has a single index area, but if specified, it must be specified as IXX. If the database has a Multi-Dataset Index, however, the keyword is required and must be set to a single index area, that is, the name of one of the index areas (IXX or I*nn*, where *nn* can be 00 to 99).

Valid Entries:

IXX or I00 to I99

Default Value:

(No default)

When you execute the EXTEND function, you receive the operator console message:

Verify that the dtfname and file-name displayed are the names associated with the file you want to extend. Issue the following command, substituting the partition number for *n*:

```
n DELETE
```

Your file is not to be deleted. It is to be extended with the newly allocated space.

Example JCL

The following shows the command to extend data area DEM for database 001 and to format the newly extended space.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL DEM001,'db001.dem',9999,DA                 Area DEM for DBID 1
 // EXTENT SYSnnn,volser,,0,rel trk/blk,number trks/blks
 // EXTENT SYSnnn,volser,,1,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
          EXTEND  AREA=DEM,DBID=1
 /*
 /&
 * $$ EOJ
```

Sample Report: EXTEND

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             EXTEND AREA=DEM,DBID=1

                    FUNCTION=EXTEND
       ❷               AREA=DEM
                       DBID=1


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX



                      DEM - STATUS INFORMATION
                    {  DEVICE TYPE -    3380
                    {  BLOCK LENGTH -  1,024
                    {  BLOCKS/TRACK -     31
                    {
            ❹       {  BLOCKS IN USE -        13
                    {  BLOCKS UNUSED -       142
                    {  TOTAL BLOCKS -        155
                    {  TOTAL TRACKS -          5
                    {  PERCENT FULL -          9
```

❹

Report of space allocated and used in the area after the EXTEND has completed.

If a system failure has occurred, CA Datacom/DB may be unable to accurately determine the number of "Blocks in Use" and "Blocks Unused." When this occurs, CA Datacom/DB reports blanks in these fields as well as in the "Percent Full" field.

Chapter 14: EXTRACT (Extract Data Table)

The EXTRACT function of DBUTLTY reads a CA Datacom/DB table and writes it to the output data set in the specified sequence. EXTRACT outputs data in expanded form.

See COMM STATUS (Print Status of Multi-User Jobs) (see page 180) for information about DBUTLTY STATUS console command support for this function.

For information about the impact of table partitioning on the EXTRACT function, see The EXTRACT Utility (see page 41).

EXTRACT

When to Use

Use the EXTRACT function when you intend to process table information for use with the CA Datacom/DB Record Migration Facility utility (DDRMFLM) in fixed-block format or for use outside the CA Datacom environment. To back up a table in compressed form, you must use the BACKUP function for the entire data area that contains the table.

Extracting in Native Key Sequence

An extract of a table in Native Key sequence without MULTUSE=YES requires that the database not be open for update by any copy of CA Datacom/DB. This protection ensures the successful retrieval of the data. Multiple EXTRACTs may be executing at once with the same or different tables in the database. However, if two or more are executing within a single area, the completion of the first updates the Directory (CXX) to remove the fact that the area is being unloaded, which loses protection for any remaining EXTRACTs for an area. Therefore, to ensure protection, only one table in an area should be done at one time.

An extract with MULTUSE=YES requires that the database be set in the Multi-User Facility startup without ACCESS NOOPT. The database is opened for update during the EXTRACT, and that update protects the area containing the table to ensure a successful retrieval of the data.

Note: With MULTUSE=YES, only one EXTRACT or BACKUP may be executing at one time within a single data area.

Extracting in Physical Key Sequence

With UPDATE=YES, the physical extract occurs with the rules as previously described (for Native Key sequence) without MULTUSE=YES.

Without UPDATE=YES, the physical extract executes with few requirements and no protection. It was designed for the special case of extracting the data with I/O errors or during maintenance for a *hot* backup. With no protection, the data is the best that it is possible to get by reading what is on the DASD at the time it is read. This option should be executed when the submitter is watching or controlling other activity.

EXTRACT Output

The output contains all the records for the table in the sequence specified by the SEQ keyword. If the table is empty, the function creates a valid null output data set.

Usually, EXTRACT writes each data record as a fixed-length record in a fixed-block format. When you request an EXTRACT of a table that was loaded from a variable-length VSAM file for use by the CA Datacom VSAM Transparency, the function writes each data record as a variable-length record in a variable block format.

Master List Requirements

The data area EXTRACT function uses a form of GSETL/GETIT or GSETP/GETPS command for reading the data blocks. This includes reading multiple blocks per I/O. You can allocate sequential buffers to use for reading the area by specifying the SEQBUFS= keyword.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO and either SEQ=NAT or SEQ=PHY,UPDATE=YES*

- The database may *not* be open for update anywhere.
- The database may or may not be open in any Multi-User Facility for read.
- The database may or may not have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO and either SEQ=NAT or SEQ=PHY,UPDATE=YES*

- The database will be opened for *unloading* status.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when SEQ=PHY,UPDATE=NO*

- The database may or may not be open for update or read or *unloading* anywhere.
- The database will not be opened for read or update or *unloading.*
- This option has no requirements, it provides no protection.

Environmental **Requirements** *when MULTUSE=YES*

- The database may or may not be open for update in the connected Multi-User Facility
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/READ/UTLTY.
- The area ACCESS must be set WRITE/READ/UTLTY.
- The area may have no table in an open URT with UPDATE=YES.
- The area may *not* have any other MULTUSE=YES utility executing.

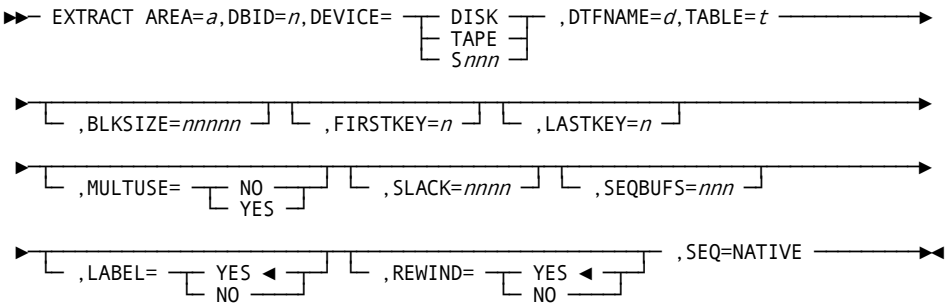
Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The database will have no *unloading* status set.
- The utility sets ACCESS area READ (utility set).
- The utility sets EXTRACT executing this Multi-User Facility this database or area.

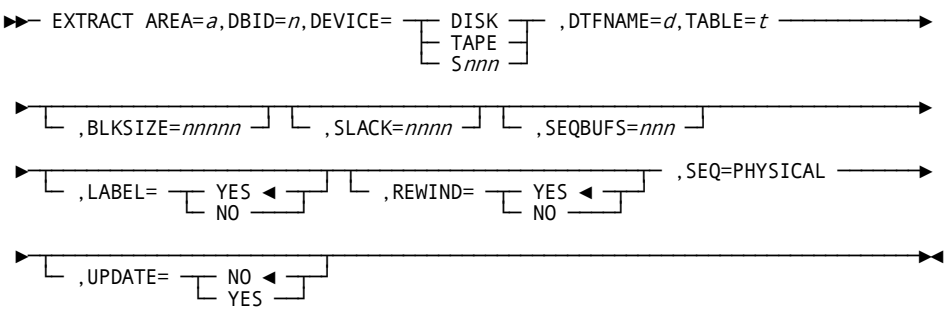
How to Use

You can execute this command in either Single User or Multi-User mode. Execute the EXTRACT function using the following command format:

Extract Data Table Native (VSE)



Extract Data Table Physical (VSE)



Command

EXTRACT

Invokes the function to extract data in expanded form.

Required Keywords

AREA=

Identifies the data area containing the table.

Code *** (three asterisks) to use the Full Parent name, that is, if you are extracting a partition Full Parent table.

Valid Entries:

DATACOM-NAME of the area, or *** to extract a partition Full Parent table

Default Value:

(No default)

,DBID=

Identifies the database for the area to be read.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

DEVICE=

Specifies the device type for the output file.

When TAPE is being used, the system logical unit (SYS) number must be 10 (SYS010).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the output data set. If DEVICE=TAPE, this name must match the name which you code in your z/VSE JCL TLBL statement, otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Note: The name may not be the standard DTFNAME for the area containing the table, which is the area name followed by the DBID in 3 characters, if less than 1000, or 4 characters, if 1000 or larger.

Default Value:

(No default)

,TABLE=

Specifies the name of an individual table to be extracted.

Valid Entries:

DATACOM-NAME of the table

Default Value:

(No default)

Optional Keywords

,BLKSIZE=

Indicates the block size for the output data set. The number supplied with this keyword overrides the block size supplied in your JCL. If block size is specified in the JCL and this keyword is not used, the JCL block size is used. In the absence of either this parameter or a specification in the JCL, the utility uses a default as specified below.

Valid Entries:

For fixed-length tables: An integer value that is a multiple of the table record length.

For variable-length tables (CA Datacom VSAM Transparency): An integer value that is at least the length of the longest record plus 8.

Default Value:

CA Datacom/DB selects the largest multiple of the record size which is less than or equal to 4096, if the DEVICE=DISK, or 32767, if the DEVICE=TAPE, (for a fixed-length record).

For example, if the DEVICE=DISK and the record size is 100, the block size is computed as 4000, or if the record size is 105, the block size is computed as 4095.

,FIRSTKEY= and/or ,LASTKEY=

These keywords allow you to select a segment of a table or area by Native Key (SEQ=NATIVE) value to be processed for EXTRACT. The EXTRACT function normally defaults to include the full Native Key range from low values to high values. FIRSTKEY= overrides the default starting position while LASTKEY= overrides the default ending position, limiting the records retrieved for output to the EXTRACT file. If one keyword is specified and the other is not specified, a default value is selected for the missing keyword. Normal CA Datacom key lengths are from 1 through 180 bytes. The key value data entered for FIRSTKEY= and LASTKEY= can be from 0 (zero) through 59 bytes long. You only need to code the number of positions of the key value that are significant to you, but be aware that the value coded is *left justified regardless of key field data type.* The utlity pads the low order or remaining positions of the key value with low values on FIRSTKEY= and high values on LASTKEY=. To extract the entire area in one step, specify neither FIRSTKEY= nor LASTKEY=. Alternately, you could specify FIRSTKEY=00 and/or LASTKEY=00 which causes processing to start at low values and end at high values. The length of zero indicates that the padded key values (low values and high values) are to be used.

The data is not edited or interpreted. Therefore, if the key value data you need to enter is in binary, you must set up the JCL using a hexadecimal mode display and enter the key value data in hexadecimal. This data can contain blanks, commas, and any other special characters. CA Datacom/DB cannot edit this data for syntax, so the control statement can *look* invalid, especially when it includes blanks or commas or key value data containing English words. For example, FIRSTKEY=12121212121212 means the length is 12, key value is 121212121212. Another example, FIRSTKEY=10FIRSTKEY10 means the length is 10, key value is FIRSTKEY10.

Valid Entries:

A 2-byte length (in the range of 00 through 59) followed by a character string of key value bytes (for the length specified) that represents the key value.

Default Value:

The starting/ending position of the full Native Key range from low values to high values.

LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), LABEL= specifies whether tape labels are to be processed. We recommended that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,MULTUSE=

(For area level DBUTLTY control only.) If you specify MULTUSE=, do not specify SEQ=PHYSICAL.

Note: EXTRACT with a Full Parent table must *not* specify MULTUSE=YES.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

NO or YES

Default Value:

NO

REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

,SEQ=

Indicates the sequence for the output data.

NATIVE

CA Datacom/DB reads the Index and writes the data in logical Native Key sequence. When using SEQ=NATIVE, you can use the FIRSTKEY=/LASTKEY= optional keywords to select only a portion of the table or area. extracts.

PHYSICAL

CA Datacom/DB does not read the Index and outputs the data in physical track sequence.

Note: Do not specify SEQ=PHYSICAL if you specify MULTUSE=.

EXTRACT with a Full Parent table must *not* specify MULTUSE=YES.

Valid Entries:

NATIVE or PHYSICAL

Default Value:

NATIVE

,SEQBUFS=

If specified, SEQBUFS= must range from 0-256, even numbers only. We recommended for best performance to specify SEQBUFS=128 or SEQBUFS=256. If memory is limited and slower performance is acceptable, lower numbers should be used.

If MULTUSE= is omitted or specified as NO, the default is SEQBUFS=8. Any value specified for SEQBUFS= less than 8 is forced up to 8. Any value greater than 128 is forced to 128. If the number of data buffers (specified with the DATANO= parameter in the DBMSTLST macro) used by this DBUTLTY was specified as 192 or more, those data buffers are used and no additional memory is allocated. The SEQBUFS= value is edited and ignored (for best performance) without a larger DATANO= parameter value specification. The DATANO= buffers are also used if they are equal to or greater than the SEQBUFS= value multipled by 1.5. Otherwise, the number of SEQBUFS= is allocated as additional sequential buffers with additional memory required.

If MULTUSE=YES is specified, the default is SEQBUFS=128. If SEQBUFS= is specified as 0-8, it is rounded up to 8. If SEQBUFS= is specified as 130-134, it is rounded up to 136, in which case no read-ahead nor multi-block reads occur. If SEQBUFS= is specified as 2-128 (even numbers only), that many private sequential buffers are allocated (memory must be available) in the MUF address space. Read ahead is done. Multiple blocks are read with each I/O when possible to reduce elapsed times and CPU times with the number per I/O as one-half of the SEQBUFS. If set as 130-256, no private sequential buffers are allocated, normal existing data buffers are used in the MUF address space. This function shares those buffers with all other tasks using MUF. If set as 132-256, read ahead is done, and multiple blocks are read with each I/O to reduce elapsed times and CPU times with the number per I/O as the SEQBUFS number minus 128 and one-half of that result. The maximum blocks per read I/O is always 64.

Valid Entries:

0-256 (even numbers only)

Default Value:

128

,SLACK=

Provides record padding in the output data. This allows a larger size to be written to the output file to assist in possible record expansion. Without this parameter, the data written to the output file is in the size used by CA Datacom/DB. The pad, if any, is always blanks.

Valid Entries:

0 to 9999

Default Value:

0

,UPDATE=

Indicates whether the database can be open for update at the start or during the backup.

Note: Only valid if you specify PHYSICAL for the SEQ= keyword.

NO

Allows tables in the database to be updated and allows the INIT or LOAD functions on the data area during the backup.

YES

Specifies that CA Datacom/DB determines that the data area is loaded with a loaded Index and that the database is open for read-only with the unloading indicator set.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL (Output to Tape)

The following shows the command to extract the data table POL in area DEM in database 1 to the output data set, OUT1DD, on tape. CA Datacom/DB outputs the data in physical track sequence.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL DEM001,'db001.dem',,DA                     Area DEM in DBID 1
 // EXTENT SYSnnn,volser
 // ASSGN SYS010,TAPE
 // PAUSE MOUNT TAPE
 // TLBL OUT1DD,'out1dd'                            Output data
 // EXEC DBUTLTY,SIZE=750K
          EXTRACT  AREA=DEM,DBID=1,DTFNAME=OUT1DD,TABLE=POL,
            DEVICE=TAPE,SEQ=PHYSICAL
 /*
 /&
 * $$ EOJ
```

Example JCL (Output to Disk)

The following shows the command to extract the information for the table PAY in physical sequence to OUT1 on disk.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL PAY001,'db001.dem',,DA                     Area PAY in DBID 1
 // EXTENT SYSnnn,volser
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL OUT1,'out',1,SD                            Output data
 // EXTENT,volser,1,0,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
          EXTRACT  AREA=PAY,DBID=1,TABLE=PAY,DTFNAME=OUT1,DEVICE=DISK,
            BLKSIZE=800,SEQ=PHYSICAL
 /*
 /&
 * $$ EOJ
```

Sample Report: EXTRACT

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             EXTRACT AREA=DEM,DBID=1,DTFNAME=OUT1DD,DEVICE=TAPE,TABLE=POL,SEQ=PHYSICAL

                    FUNCTION=EXTRACT
       ❷               AREA=DEM
                       DBID=1
                       DEVICE=TAPE
                       DTFNAME=OUT1DD
                       SEQ=PHYSICAL
                       TABLE=POL

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                     TABLE          RECORDS

                     PNC                 0
                     PNM                 0
                     POH                 0
     ❹               POL                 4

                     TOTAL               4
```

❹

Confirmation that four records were extracted from the requested table.

Chapter 15: FILEIN (Concatenate Input)

The FILEIN function is used to do one load with data from several backups. You can specify multiple FILEIN commands in a single job.

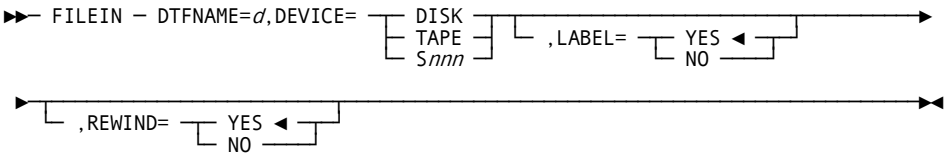
FILEIN

When to Use

Use FILEIN when you want to do one load using data from several backups. It may be used in front of a LOAD or a REPLACE.

How to Use

Multi-User mode is required to execute this command. Execute the FILEIN function using the following command format.



Command

FILEIN

FILEIN invokes the function to concatenate multiple backups to do one load with all the data.

Required Keywords

DEVICE=

DEVICE= specifies the device type for the input data set.

When DEVICE=TAPE is specified, the system logical unit (SYS) number must be 5 (SYS005).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

DTFNAME=

DTFNAME= specifies the z/VSE JCL TLBL/DLBL name for the input data set.

If DEVICE=TAPE, this name must match the name that you code in your z/VSE JCL TLBL statement. Otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

Optional Keywords

,LABEL=

LABEL= specifies, if DEVICE=TAPE (that is, either TAPE or the S*nnn* form), whether tape labels are processed. We recommend that you always use labeled tapes.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,REWIND=

REWIND= is used, if DEVICE=TAPE (that is, either TAPE or the S*nnn* form), can be specified as YES to indicate that the tape is to be rewound before OPEN and after CLOSE.

NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

Example JCL

Following is example JCL showing the use of the FILEIN statement with the REPLACE function.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL BKPUP1,'input'                          Input data set
 // EXTENT ,volser
 // DLBL BACKUP,'input'                          Input data set
 // EXTENT ,volser
 // ASSGN SYS001,device type,VOL=volser,SHR
 // ASSGN SYS002,device type,VOL=volser,SHR
 // DLBL SORTWK1
 // EXTENT SYS001,volser,,,rel trk/blk,number trks/blks
 // DLBL SORTWK2
 // EXTENT SYS002,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=768K
                      FILEIN  DEVICE=DISK,DTFNAME=BKPUP1
                      REPLACE DBID=1,TABLE=F01,DTFNAME=BACKUP,SLACK=12,SORT=100,
                         SORTWK=2,OPTION1='PRINT=ALL',DEVICE=DISK,MULTUSE=YES

 /*
 /&
 * $$ EOJ
```

Sample Report

Following begins a sample report showing the use of the FILEIN statement with the REPLACE function.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          FILEIN  DEVICE=DISK,DTFNAME=BKPUP1

                    FUNCTION=FILEIN
          ❷            DEVICE=DISK
                       DTFNAME=BKPUP1
          ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          REPLACE DBID=1,TABLE=F01,DTFNAME=BACKUP,SLACK=12,SORT=100,
                               SORTWK=2,OPTION1='PRINT=ALL',DEVICE=DISK,MULTUSE=YES

                    FUNCTION=REPLACE
          ❷           DBID=00001
                       DEVICE=DISK
                       DTFNAME=BACKUP
                       MULTUSE=YES
                       OPTION1=PRINT=ALL
                       SLACK=0012
                       SORT=000000100
                       SORTWK=02
                       TABLE=F01
          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX

         ❹          TABLE          RECORDS

                    F01               50
```

❹

Table and records information.

Chapter 16: FILEOUT (Define Output Data Sets)

The FILEOUT function is used to define output data sets, for example to define one or more output data sets used by the EXTBKUP function (see EXTBKUP (Extract from Backup File) (see page 305)). If the table is empty, FILEOUT creates a valid null output data set.

Note: When used to define one or more output data sets used by the EXTBKUP function, all FILEOUT statements must precede the EXTBKUP statement.

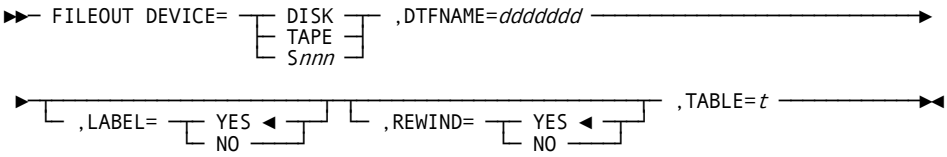
FILEOUT

When to Use

Use the FILEOUT function when you want to define output data sets.

How to Use

Multi-User mode is required to execute this command. Execute the FILEOUT function using the following command format:



Command

FILEOUT

Invokes the function to define output data sets.

Required Keywords

Specifies the name of the first/next extract output file.

Each table specified in a FILEOUT must be found in the control records for the input backup file. If any of the specified tables are not found in the control records for the input backup file, a message in the report indicates the invalid table name, and the utility terminates as a failure.

,DTFNAME=

Specifies the name of the first/next extract output file.

Valid Entries:

A valid extract output file name.

Default Value:

(No default)

DEVICE=

Specifies the device type, either DISK, TAPE, or S*nnn.*

DEVICE=S*nnn* allows you to use the *nnn* to specify a different SYS number (for tape only).

When you specify DEVICE=DISK, CA Datacom/DB finds the optimum block size less than 27998.

When you specify DEVICE=TAPE, the system logical unit (SYS) number must be 10 (SYS010). With DEVICE=TAPE the block size is within 32766.

Valid Entries:

DISK, TAPE, or S*nnn*

Default Value:

(No default)

,TABLE=

Specifies the name of an individual table from which to define the output data set.

Each table specified in a FILEOUT must be found in the control records for the input backup file. If any of the specified tables are not found in the control records for the input backup file, a message in the report indicates the invalid table name, and the utility terminates as a failure.

Valid Entries:

DATACOM-NAME of the table

Default Value:

(No default)

Optional Keywords

,LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), specifies whether tape labels are to be processed. We recommend that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,REWIND=

Specifies action at OPEN and CLOSE. If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

Examples

For sample JCL showing the use of the FILEOUT statement in the EXTBKUP function, see Example JCL (see page 309).

For a sample EXTBKUP report showing the use of the FILEOUT statement, see Sample Report: EXTBKUP (see page 310).

Example JCL

Following is a JCL example showing the use of a FILEOUT statement with the EXTBKUP function.

The following shows the command to extract the data table POL in area POL in database 1 to the output data set, POLOUT. The FILEOUT statement is shown immediately before the EXTBKUP line.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the note above and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL INFILE,'db001.pol',,DA                 Input file
 // EXTENT SYSnnn,volser
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL POLOUT,'polout'                        Output data
 // EXEC DBUTLTY,SIZE=750K
          FILEOUT  DTFNAME=POLOUT,TABLE=POL,DEVICE=S005
          EXTBKUP  DEVICE=DISK,DTFNAME=INFILE
 /*
 /&
 * $$ EOJ
```

Sample Report: FILEOUT

Following is a sample FILEOUT report.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             FILEOUT DTFNAME=POLOUT,TABLE=POL

                    FUNCTION=FILEOUT
       ❷               DTFNAME=POLOUT
                       TABLE=POL

     .
     .
     .
       ❸
 BACKUP FILE INFORMATION
 JOBNAME  RUN UNIT     DATE       TIME     CXXNAME     DBID    SEQ    RECID   AREA
 BKUPJOB    17,097   02/14/2003  07.55.02  SPRMUFF      797    PHY     NO     A01
       ❹
 OUTPUT  TABLE INFORMATION
 A01    C01   101      30,666
 A01    C02   102      47,792
 A01    F01   103      41,167
 A01    F02   104      45,567
 A01    F03   105      41,000
 A01    RNA   106      40,776
 A01    DNA   107      45,500
 A01    NRA   110      45,292
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Backup file information.

❹

Output table information.

Chapter 17: FLEXPOOL (Dynamic Buffers)

The FLEXPOOL function dynamically adds or deletes data area or Index Area (DXX or IXX) flexible buffers to the system. The flexible part is that, with the FLEXPOOL function, these buffers can be removed from the system (normal buffers cannot). Therefore, for example, you can use the Multi-User startup option FLEXPOOL to add 500 data buffers and later reduce that number with the DBUTLTY FLEXPOOL function. You can also use the DBUTLTY FLEXPOOL function to add flexible pool buffers up to the maximum combined total for the Multi-User startup option and the DBUTLTY function (and the DATAPOOL or SYSPOOL buffers) of 99999 for each class.

FLEXPOOL

The FLEXPOOL function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

When to Use

Use FLEXPOOL to add or delete flexible buffers.

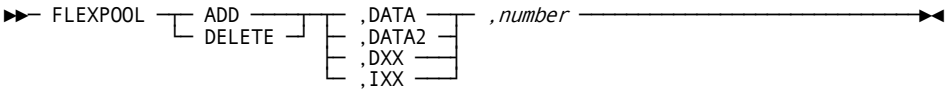
To see the current flexible buffer counts, use a FLEXPOOL add or delete statement for the desired buffer class with 0 specified. The function generates a message, on the console or on the DBUTLTY report, indicating the status of the request and the number of buffers added or deleted, remaining, and totals.

Note: The total buffers by type are those buffers for general use. They do not include any special data buffers added for sequential processing.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

FLEXPOOL

Invokes the function for dynamic buffer maintenance.

Required Keywords

ADD

Specifies to add buffers.

DELETE

Specifies to delete buffers.

,DATA

Indicates data area primary buffers.

,DATA2

Indicates data area secondary buffers, larger buffers than the primary buffers.

,DXX

Indicates low-level Index Area buffers.

,IXX

Indicates high-level Index Area buffers.

,number

Specify the number of buffers of the type indicated that are to be added or deleted.

Valid Entries:

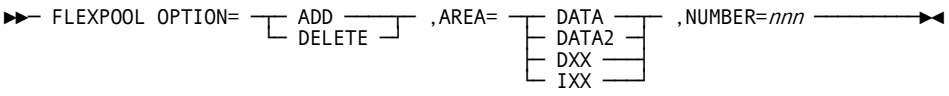
0 to 9999

Default Value:

(No default)

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. To use this function, execute CA Datacom/DB Utility (DBUTLTY) using the command:



Command

FLEXPOOL

Invokes the function for dynamic buffer maintenance.

Required Keywords

OPTION=

Specifies the type of activity to be performed.

Valid Entries:

ADD or DELETE

Default Value:

(No default)

,AREA=

Specifies the type of buffer to be added or deleted.

DATA

Indicates data area primary buffers.

DATA2

Indicates data area secondary buffers, larger buffers than the primary buffers.

DXX

Indicates low-level Index Area buffers.

IXX

Indicates high-level Index Area buffers.

Valid Entries:

DATA, DATA2, DXX, or IXX

Default Value:

(No default)

,NUMBER=*nnn*

Specify the number of buffers of the type indicated that are to be added or deleted. If the amount of memory does not allow the total number requested to be added, as many as possible are added. If the number requested to be deleted is more than exists for the type of buffer, as many as exist are deleted.

Valid Entries:

0 to 9999

Default Value:

(No default)

Example JCL

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          FLEXPOOL  OPTION=ADD,AREA=DATA,NUMBER=20
 /*
 /&
 * $$ EOJ
```

Sample Report: FLEXPOOL

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             FLEXPOOL OPTION=ADD,AREA=DATA,NUMBER=20

                    FUNCTION=FLEXPOOL
       ❷               OPTION=ADD
                       AREA=DATA
                       NUMBER=20

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


                    DB00606I - FLEXPOOL ADD    DATA 00020, NOW TOTAL 00220 FLEXPOOL 00020

                                        ❹      ❺     ❻              ❼            ❽
```

❹

Indicates the ADD or DELETE function.

❺

Indicates the buffer class (data, IXX, or DXX).

❻

Indicates how many buffers were processed during this request. That is, if the request was to add 1000 but there is only enough memory for 700, this value is 700. Also, if FLEXPOOL had 700 and a delete request was made for 9999, this value is 700.

❼

Indicates the total number of buffers in this class (includes the Multi-User startup options, DATAPOOL and SYSPOOL, plus any FLEXPOOL buffers).

❽

Includes the current number of FLEXPOOL buffers.

Chapter 18: INIT CXX/FXX/IXX/LXX/Data Area

Each option is described in a separate section of this chapter. You must run the INIT function to format space allocated to these areas before the area can be used by CA Datacom/DB. You do not need to format the Statistics and Diagnostics Area (PXX).

Restrictions

You can initialize a database Index **only** if no task is actively processing the database. You can use the ACCESS option. To prevent opens for a database that contains the index that you want to initialize, see ACCESS (Restricting Opens or Maintenance) (see page 55).

A data area or index area data set cannot be shared by two CXXs. This can result in data integrity problems and index corruption.

Multivolume INITs

You can initialize these areas across multiple extents on multiple volumes. For a description of how the operating system handles secondary allocations, see your operating system JCL manual.

INIT CXX (Format Directory)

The INIT CXX option formats the Directory (CXX) data set. This function clears all of the blocks in the data set. You must select an internal name for the Directory. If you use multiple Directories, you should name the Directories uniquely so that you can identify which Directory is running when functions are executed.

When to Use

Use the INIT CXX option when setting up a new Directory, loading a full backup of the Directory, or converting to a new CA Datacom/DB version.

Restriction

Make certain that the MUF is down before initializing the CXX. If the MUF is up, bring it down with an EOJ before initializing the CXX. You can bring the MUF back up after the CXX has been initialized.

CXX Name and Format

You must specify an identifier for the Directory when it is initialized. The utility functions display the identifier whenever you request maintenance or reporting on the Directory.

The CXXNAME is used to control access through external security. It is also used to protect which Index and data areas are linked or connected to the correct Directory. **All Directories on your system should be uniquely named.**

The Directory is formatted with fixed 4096-byte blocks.

Identifying the Directory

You must identify the Directory as either a CA Datacom/DB Directory or a CA Datacom/AD Directory. You can load a CA Datacom/AD Directory only with a backup of a CA Datacom/AD Directory. You can load a CA Datacom/DB Directory from a CA Datacom/DB Directory and a CA Datacom/AD backup.

How to Use

You can execute this command in either Single User or Multi-User mode. Execute the INIT function with the following command format:

Command

INIT

Invokes the function to format areas for CA Datacom/DB use.

Required Keywords

AREA=CXX

Identifies the area to be formatted as the Directory.

,CXXNAME=

Assigns an internal name to the new Directory to be created.

Valid Entries:

A 1- to 8-character name

Default Value:

(No default)

,DATACOM=

Identifies the new Directory as a CA Datacom/DB Directory or a CA Datacom/AD Directory.

Note: CA Datacom TR sites should specify AD.

Valid Entries:

AD or DB

Default Value:

(No default)

Example JCL

The following shows the command to initialize the Directory (CXX).

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
// ASSGN SYSnnn,device-type,VOL=volser,SHR
// DLBL CXX,'cxx',9999,DA                        Directory data set
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// EXEC DBUTLTY,SIZE=750K
         INIT     AREA=CXX,CXXNAME=PRODCXX,DATACOM=DB
/*
/&
* $$ EOJ
```

Sample Report: INIT CXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             INIT AREA=CXX,CXXNAME=PRODCXX,DATACOM=DB

                    FUNCTION=INIT
       ❷               AREA=CXX
                       CXXNAME=PRODCXX
                       DATACOM=DB

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************




                      CXX - STATUS INFORMATION

                    { DEVICE TYPE -    3380
                    { BLOCK LENGTH -  4,096
                    { BLOCKS/TRACK -     10
                    {
            ❹       { BLOCKS IN USE -         2
                    { BLOCKS UNUSED -     2,998
                    { TOTAL BLOCKS -      3,000
                    { TOTAL TRACKS -        300
                    { PERCENT FULL -          1
```

❹

Messages indicating the status of the Directory at successful completion of the initialization.

INIT Data Area (Format Data Area)

The INIT Data Area option formats the data area. It updates the first block with control information. This first block is subsequently updated when maintenance is performed. The block size used is the size specified for the area in the Directory (CXX) definition.

Performing an INIT on a data area has no performance benefit. It should only be done before the first load, and then only if needed. An INIT of an Index or data area clears the utility history for that area (see Type H - Utility Function History Report (see page 659)).

Note: When you specify VERIFY=YES, information in the control block is preserved instead of being lost. This history save and restore process presumes a successful completion and does not occur on all error conditions. For more information, see the description of VERIFY= in the table that begins on page How to Use.

When to Use

Use this function to allocate a new data area. A data area must be initialized before CA Datacom/DB can use it. This is necessary only once for any data area unless you modify the block size to accommodate changes.

Data Set Name

CA Datacom/DB saves the data set name of the area in the Directory at the time of the INIT. The data set name is then available for reporting. It is also available for dynamic allocation volumes, depending on operating system requirements and operating system maximums. If initializing an existing data set (DISP=SHR or DISP=OLD), CA Datacom/DB formats all currently allocated tracks for use. If initializing a new data set (DISP=NEW), CA Datacom/DB requests and formats the primary allocation only if one volume is provided. If multiple volumes are provided, CA Datacom/DB requests secondary allocations until space has been allocated on the number of volumes specified. For a description of how the operating system handles secondary allocations, see your operating system JCL manual.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any Multi-User Facility for read.
- The database may *not* have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO*

- The database will be opened for update.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when MULTUSE=YES*

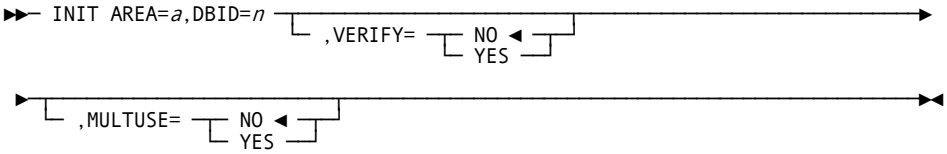
- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The area may have no table in an open User Requirements Table.
- The area may *not* have any other MULTUSE=YES utility executing.
- The area must be closed.

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets INIT executing this Multi-User Facility this database or area.

How to Use

You can execute this command in either Single User or Multi-User mode. Execute the DBUTLTY function using the following command format:



Command

INIT

Invokes the function to format areas for CA Datacom/DB use.

Required Keywords

AREA=

Identifies the data area to be formatted.

Valid Entries:

DATACOM-NAME of the area in the specified database

Default Value:

(No default)

,DBID=

Identifies the database containing the area that is to be initialized.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Optional Keywords

,MULTUSE=

(For area level DBUTLTY control only.) In support of area level DBUTLTY control, specifying MULTUSE=YES in the INIT function for a data area allows a single area in a database to be initialized with no disabling of the other areas in the database. When MULTUSE=YES, the function executes in a format that is a combination of Multi-User Facility and Single User modes. The database involved must be able to be opened for update in the Multi-User Facility address space. The area, however, must have no tables open. Once begun, the function does not allow other tasks to open any tables in the area being processed. Other tables in the database, but not in the area, may be open for read or update in the same Multi-User Facility. The actual data set open is performed in the utility region, and the JCL requirements are applied to the utility region.

When you specify or allow the default of MULTUSE=NO in the INIT function for data areas, the function executes completely in Single User mode.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

NO or YES

Default Value:

NO

,VERIFY=

If you do not specify the VERIFY= option, or if you specify VERIFY=NO (the default), an INIT opens the data set for output, builds a *new* control block, and initializes the extents.

If you specify VERIFY=YES, an INIT opens the data set for update, reads and updates an *existing* control block, and initializes the extents.

Because a new control block is always built with VERIFY=NO, specifying VERIFY=YES when the INIT is not for a new data set brings several benefits, including the following:

- Because an existing control block is read and updated instead of a new one being built, the data area history (in the control block) documenting activities such as backups and loads is not lost. Data area history information can be a valuable aid when troubleshooting various problems.
- VERIFY=YES prevents the unintentional overlaying of important information because of a JCL error, such as writing over files in PRODuction status that should have been in TEST status, or the inadvertant overlaying of the Directory (CXX) or Log Area (LXX).

Note: Even with VERIFY=YES specified, the last extent and the number of extents is lost during an initialize.

You can specify both VERIFY=YES and DISP=NEW, but if you specify both the VERIFY=YES is ignored. (Prior to Version 12.0, you were not allowed to specify YES for both VERIFY= and DISP=.)

Valid Entries:

YES or NO

Default Value:

NO

Example JCL

The following shows the command to initialize the DEM area for database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL DEM001,'db001.dem',9999,DA                 Area DEM in DBID 1
 // EXTENT SYSnnn,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
           INIT     AREA=DEM,DBID=1
 /*
 /&
 * $$ EOJ
```

Sample Report: INIT Data Area

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             INIT AREA=DEM,DBID=1

                    FUNCTION=INIT
       ❷               AREA=DEM
                       DBID=1


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX




                      DEM - STATUS INFORMATION

                    { DEVICE TYPE -    3380
                    { BLOCK LENGTH -  1,024
                    { BLOCKS/TRACK -     31
                    {
              ❹     { BLOCKS IN USE -         1
                    { BLOCKS UNUSED -       154
                    { TOTAL BLOCKS -        155
                    { TOTAL TRACKS -          5
                    { PERCENT FULL -          1
```

❹

Messages indicating status of the data area at successful completion of the initialization.

INIT IXX (Format Index Area)

The INIT IXX option formats the Index Area (IXX) data set blocks. It updates the first block with control information. This first block is subsequently updated when maintenance is performed. The default block size is 4096 for all devices, except for 3380 devices which have a block size of 3860.

Note: When you specify YES for the optional keyword VERIFY=, information in the control block is preserved instead of being lost. For more information, see the description of VERIFY= in the table that begins on page How to Use (see page 364) for more information on VERIFY=.

When to Use

Use this option to initialize a newly allocated Index Area data set. You can also use this option to clear an Index Area before reloading a database or retrieving the Index entries for all areas in a database. See LOAD (Build a Data Area) (see page 393) and RETIX (Reconstruct Index) (see page 673) for details.

An Index Area must be initialized before it can be used by CA Datacom/DB. This is necessary only once for any Index Area. However, in certain cases a reload of a database or a reconstruct index (RETIX) operation can execute more quickly, if an existing Index Area is first initialized. See LOAD (Build a Data Area) (see page 393) and the *CA Datacom/DB Database and System Administration Guide* for further details about initializing the Index Area.

An INIT of an Index or data area clears the utility history for that area (see Type H - Utility Function History Report (see page 659)).

Special Rules for Databases with Multi-Dataset Index Areas

If a database is cataloged with a definition containing Multi-Dataset Index areas, the first requirement is to initialize the IXX index area, if it is not currently initialized, including AREA=IXX and IXXAREA=IXX (or IXXAREA=***). The IXXAREA=*** specifies that every index area is to be initialized starting with the IXX. If initialized it must be able to be opened. With the IXX prepared, the next requirement is to initialize every other index area that is not initialized, then to ensure they are able to be opened.

Data Set Name

CA Datacom/DB saves the data set name of the area in the Directory at the time of the INIT. The data set name is then available for reporting. It is also available for dynamic allocation.

Restrictions

You can initialize a database Index Area only if no task is actively processing the database. To prevent opens for a database that contains an area you want to initialize, depending on operating system requirements and operating system maximums, see ACCESS (Restricting Opens or Maintenance) (see page 55). If initializing an existing data set (DISP=SHR or DISP=OLD), CA Datacom/DB formats all current allocated tracks for use. If initializing a new data set (DISP=NEW), CA Datacom/DB requests and formats the primary allocation only if one volume is provided. If multiple volumes are provided, it requests secondary allocations until space has been allocated on the number of volumes specified. For a description of how the operating system handles secondary allocations, see your operating system JCL manual.

Controlling the Index Block Size

The block size of each database Index Area (IXX) is determined during IXX initialization. When you initialize the IXX, you can override the default index block size by using the optional BLKSIZE= keyword. BLKSIZE= can be used to specify an index block size of from 4096 bytes through 8192 bytes. The BLKSIZE= value remains in effect until the next initialization of the IXX. You must pick an optimal size for your device type, in a manner similar to how the data area block size is selected.

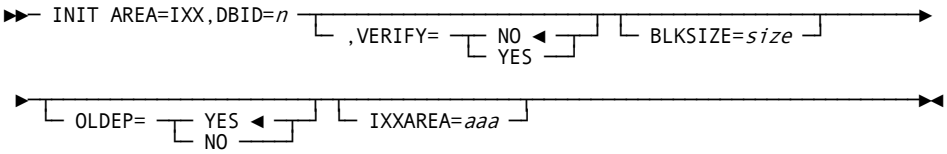
Important! We do not recommend using a block size other than the default. Changing the block size might help in the utilization of some parts of index storage and processing, but other parts could be hurt by the change. Therefore, changing the block size should only be done after first experimenting to discover whether the change causes better or worse results. Be aware that the index can double in size using 8K blocks when OLDEP= is not specified as NO.

The index is complex and proprietary, typically involving a variety of key lengths, degrees of compression, and the possibility of from one to many duplicate values. Different combinations could therefore increase or decrease DASD usage, and could also, independently, increase or decrease CPU usage. This is why we caution you that, if you want to try to improve resource utilization by changing the index block size from the default, experimentation is needed to find out what results a change in block size has with regard to the mix of applications, CPU availability, and DASD availability.

Note: An OPEN error occurs if an attempt is made to open a database that has an index block size larger than the index buffer size. See the information about the *ixxbfrsize* parameter of the SYSPOOL Multi-User startup option in the *CA Datacom/DB Database and System Administration Guide*.

How to Use

You can execute this command in either Single User or Multi-User mode. To format an Index Area, execute DBUTLTY using the following command format:



Command

INIT

Invokes the function to format areas for CA Datacom/DB use.

Required Keywords

AREA=IXX

Identifies the area to be formatted as the Index.

,DBID=

Identifies the database.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Optional Keywords

,VERIFY=

If you do not specify the VERIFY= option, or if you specify VERIFY=NO (the default), an INIT opens the data set for output, builds a *new* control block, and initializes the extents.

If you specify VERIFY=YES, an INIT opens the data set for update, reads and updates an *existing* control block, and initializes the extents.

Because a new control block is always built with VERIFY=NO, specifying VERIFY=YES when the INIT is not for a new data set brings several benefits, including the following:

- Because an existing control block is read and updated instead of a new one being built, the data area history (in the control block) documenting activities such as backups and loads is not lost. Data area history information can be a valuable aid when troubleshooting various problems.
- VERIFY=YES prevents the unintentional overlaying of important information because of a JCL error, such as writing over files in PRODuction status that should have been in TEST status, or the inadvertant overlaying of the Directory (CXX) or Log Area (LXX).

Note: Even with VERIFY=YES specified, the last extent and the number of extents is lost during an initialize.

Valid Entries:

YES or NO

Default Value:

NO

BLKSIZE=*size*

This optional keyword allows you to select a block size other than the default. BLKSIZE= can be used to specify an index block size of from 4096 bytes through 8192 bytes. The BLKSIZE= value remains in effect until the next initialization of the IXX. You must pick an optimal size for your device type, in a manner similar to how the data area block size is selected. See the description of BLKSIZE= in the table of required keywords, see How to Use (see page 101).

We do not recommend changing the block size from the default. See Controlling the Index Block Size (see page 364).

Note: An OPEN error occurs if an attempt is made to open a database that has an index block size larger than the index buffer size. See the information about the *ixxbfrsize* parameter of the SYSPOOL Multi-User startup option in the *CA Datacom/DB Database and System Administration Guide*.

Valid Entries:

4096 through 8192

Default Value:

4096 (or, for some device types, a little less)

OLDEP=

Refers to *old entry points.* When OLDEP=NO is specified, the DBNTRY entry point, designated as the primary entry point since the early 1980s, is the only entry point supported. For compatibility with prior versions, OLDEP=YES is the default, allowing the use of older entry points such as CBLDBMS and DATACOM. Those older entry points, however, use index blocks limited to a maximum of 255 entries. Specifying OLDEP=NO, to ensure that DBNTRY is used in Version 11.0, allows you to use index blocks that can hold a maximum of 8K entries. Therefore, if you want to use 8K index blocks, and for some performance enhancement, use OLDEP=NO.

Note: Do not initialize an index using the OLDEP= keyword until you are certain that the risk of needing to do a fallback to a prior version no longer exists. OLDEP= index initializations are not compatible with prior versions. If used in a prior version, an open error and a return code 43(43) **NO VALID INDEX** is received.

Alternately, if a program attempts using an old entry point with an index initialized as OLDEP=NO, the request fails with a return code 39(090) **CANNOT PROCESS OLD REQUEST.**

Valid Entries:

YES or NO

Default Value:

YES

IXXAREA=*aaa*

IXXAREA= is optional for databases that have a single Index Area IXX but, if specified, must be set as IXXAREA=IXX. IXXAREA= is required, however, for databases that have a Multi-Dataset Index and must be specified with one of the following: IXX, I*nn* (where *nn* can be 00 to 99), or *** where the *** (3 asterisks) means to initialize the IXX and then each additional index area in turn. The use of *** forces the same options for every area, including VERIFY=, BLKSIZE=, and OLDEP=.

Valid Entries:

IXX, I00 to I99, or ***

Default Value:

(No default)

Example JCL

The following shows the command to initialize the Index Area for database 10.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX010,'db010.index',9999,DA           Index Area
 // EXTENT SYSnnn,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
          INIT     AREA=IXX,DBID=10
 /*
 /&
 * $$ EOJ
```

Sample Report: INIT IXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             INIT AREA=IXX,DBID=10

                    FUNCTION=INIT
       ❷               AREA=IXX
                       DBID=10


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX




                      IXX - STATUS INFORMATION

                    { DEVICE TYPE -    3380
                    { BLOCK LENGTH -  3,860
                    { BLOCKS/TRACK -     11
                    {
            ❹       { BLOCKS IN USE -         2
                    { BLOCKS UNUSED -       108
                    { TOTAL BLOCKS -        110
                    { TOTAL TRACKS -         10
                    { PERCENT FULL -          2
```

❹

Status information about the Index Area at successful completion of initialization.

INIT LXX (Format Log Area)

The INIT LXX option prepares the Log Area (LXX) for use, formatting the blocks in accordance with CA standards and according to the BLKSIZE= parameter specified. The LXX must contain at least 103 blocks.

For information about using INIT AREA=LXX to specify the variable Log Area (LXX) block size feature, see Variable Log Area (LXX) Block Size (see page 370).

When to Use

Use the INIT LXX option to build a new Log Area when creating a new Multi-User Facility. The Log Area cannot be shared by Multi-User Facilities.

You must format the Log Area with the INIT function before it can be used by CA Datacom/DB. Do not attempt to process a table before completing this function.

Multi-Volume INITs

The Log Area can exist as one or more extents on one or more volumes depending on the operating system requirement and maximums. If multi-volumes are required for the LXX, a second EXTENT statement is required and must follow the IBM requirements for type DA datasets. See Example JCL for Multi-Volume Log Area (see page 374).

Initializing Existing Log Area

If you initialize an existing Log Area, any information recorded in the area is destroyed. The INIT function opens the data set for output. No edits are made by CA Datacom/DB concerning the current use of the data set. Be sure that any required data has been spilled to a Recovery File (RXX) and that no CA Datacom/DB system is actively using the area before requesting this function.

CA Datacom/DB maintains a count of Recovery Files built from a Log Area. The Recovery File number (displayed in the DB00310I log spill message) is reset to 1 after an INIT of the Log Area.

Log Block Size

Log records can span physical log blocks, especially when you have large records. Spanned log blocks can, in isolated cases, perform faster than a normal single block. However, for most cases, it performs slower and is not recommended.

Note: The use of the LONG VARCHAR data type creates rows that are as long as the block size of the containing area. The block size of the longest-block sized area containing the LONG VARCHAR data type should be used in calculating the Log Area block size.

CA Datacom/DB determines the physical block size by dividing the logical block size specified (BLKSIZE= value) by the number of physical blocks specified (BLOCKS= value). Do not specify values that result in a physical block size larger than 32760 or smaller than 16384. Regardless of how large the physical log block is defined, a maximum of 254 records can reside in a block. This is pertinent only to Log Areas that span physical log blocks.

The logical block size specified **must be** evenly divided by the number of physical blocks specified and be large enough to hold two times the record size plus space for the element list and include several hundred bytes of control information.

Variable Log Area (LXX) Block Size

The benefits of variable logging are reduced elapsed times for most update applications and reduced LXX write and read I/O. The CPU cost of using variable logging is about the same as using fixed logging. For an overview of the variable LXX block size feature, see the *CA Datacom/DB Database and System Administration Guide*.

Also related to variable logging are the following Multi-User startup options: X_LOG_AHEAD_BLKS and X_LOG_MINIMUM_BLK, as described in the *CA Datacom/DB Database and System Administration Guide*.

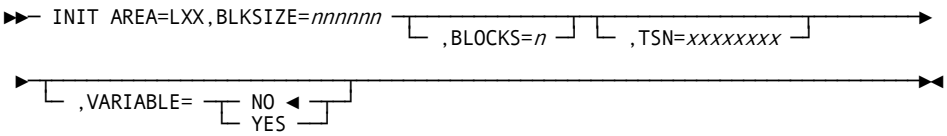
Restrictions and Requirements

Note:

- With variable logging, the block size must be set to 32760.
- This option is available for all DASD, except that in z/VSE a device defined as FBA may not be initialized as variable.
- Spanned logging cannot be used for a variable LXX.
- Variable logging is provided for the LXX. It is not supported for the Force Area (FXX). The FXX does not need to be initialized, however, when fixed logging is converted to variable logging.
- A variable LXX must be at least 250 tracks.

How to Use

You can execute this command in either Single User or Multi-User mode. Execute the INIT function with the following command format:



Command

INIT

Invokes the function to format areas for CA Datacom/DB use.

Required Keywords

AREA=LXX

Identifies the area to be formatted as the Log Area.

,BLKSIZE=

Indicates the logical block size. CA Datacom/DB uses this value and the BLOCKS= value to determine the physical block size. If the specified BLKSIZE= value is not evenly divisible by the BLOCKS= value, DBUTLTY uses the lowest multiple of the BLOCKS= value which is greater than the specified BLKSIZE= value. The resulting physical block size may not be larger than 32760. See the BLOCKS= keyword.

Note: For a variable LXX, the BLKSIZE *must* be 32760. For a fixed length LXX, a size of one-half track is commonly used.

Valid Entries:

An integer from 16384 to 98280

Default Value:

(No default)

Optional Keywords

,BLOCKS=

Specifies the number of physical blocks per logical block. BLOCKS= may *not* be 2 or 3 if the LXX is variable.

Valid Entries:

An integer from 1 to 3

Default Value:

1

,TSN=

Specifies the starting number for transaction sequence numbers (TSNs).

If you are running with more than one Multi-User Facility against the same data areas, you should specify a different starting sequence number for each Multi-User Facility to ensure that the sequence numbers are not duplicated.

The DBUTLTY REPORT AREA=LXX and RESTART reports print the highest TSN previously used.

Valid Entries:

A hexadecimal value from 00000000 to FFFFFFFF

Default Value:

00000000

,VARIABLE=

Specifies whether you want to use the variable Log Area (LXX) block size feature (for details see Variable Log Area (LXX) Block Size (see page 370)). Specify YES to use the feature. NO is the default.

Valid Entries:

YES or NO

Default Value:

NO

Example JCL for Single Log Area

The following shows the command to initialize the Log Area with a physical block size of 16384.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL LXX,'log',9999,DA                      Log Data File
 // EXTENT SYSnnn,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
          INIT     AREA=LXX,BLKSIZE=16384
 /*
 /&
 * $$ EOJ
```

Example JCL for Multi-Volume Log Area

The following shows the command to initialize the Log Area using multiple extents.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // ASSGN SYSnnn+1,device type,VOL=volser2,SHR
 // DLBL CXX,'cxx',,DA                                        Directory data set
 // EXTENT SYSnnn,volser
 // DLBL LXX,'log',9999,DA                                    Log Data File
 // EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks    Primary on 1st volser
 // EXTENT SYSnnn+1,volser2,1,1,rel trk/blk,number trks/blks Secondary on 2nd volser
 // EXEC DBUTLTY,SIZE=750K
           INIT     AREA=LXX,BLKSIZE=16384
 /*
 /&
 * $$ EOJ
```

Sample Report: INIT LXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             INIT AREA=LXX,BLKSIZE=16384

                    FUNCTION=INIT
       ❷               AREA=LXX
                      BLKSIZE=16384


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX



                    LXX - STATUS INFORMATION

                  { DEVICE TYPE -    3380
                  { BLOCK LENGTH -  8,192
                  { BLOCKS/TRACK -      5
                  {
             ❹    { BLOCKS IN USE -         1
                  { BLOCKS UNUSED -       499
                  { TOTAL BLOCKS -        500
                  { TOTAL TRACKS -        100
                  { PERCENT FULL -          0
                  { CURRENT TSN - 00000000
```

❹

Status information about the Log Area at successful completion of the initialization. (The TSN value is the transaction sequence number in hexadecimal.)

INIT FXX (Format Force Area)

The Force Area (FXX) contains logging-related information that has been *forced* from the Log Area (LXX). For details about the conditions that can cause information to be forced from the Log Area, see the chapter in the *CA Datacom/DB Database and System Administration Guide* about using logging.

When to Use

Use the INIT FXX option to build a new Force Area. You must format the Force Area with the INIT function before it can be used by CA Datacom/DB.

Initializing an Existing Force Area

If you initialize an existing Force Area, any information recorded in the area is destroyed. The INIT function opens the data set for output. No edits are made by CA Datacom/DB concerning the current use of the data set. Be sure that no CA Datacom/DB system is actively using the area before requesting this function.

Force Area Block Size

The FXX is paired with the LXX in that it must have the same physical and logical block size. The number of tracks required is dependent upon the number of tasks generated in the Multi-User Facility or MUFplex.

The FXX is divided into a control block or blocks and log record blocks. Criteria for determining the number of control blocks is based upon the number of total tasks to be supported with this Multi-User Facility or MUFplex. During initialization (INIT AREA=FXX), the TASKS= keyword is used to specify the number of tasks to be supported. See the table that begins on page How to Use (see page 376) for more information on the TASKS= keyword.

The number of control blocks that are reserved is based upon the following formula: 8192 + (tasks * 32). If running the typical 3390 block size of 27998 with less than 618 tasks, all fits in one control block. Nine control blocks would be required with 6999 tasks.

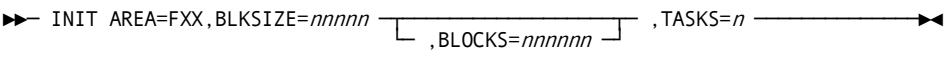
The number of blocks for log records depends upon the number of *active requests* that could possibly be forced from the LXX and upon the size of the forced log records. Active requests are forced only rarely. However, if the Multi-User Facility requires space and space is not available, all log requests fail until the Multi-User Facility is canceled and restarted. To prevent this situation from occurring, calculate the number of force blocks required to provide space for one log record per task.

An easy approach is to allow one logical block per task. While active requests are unlikely, the following conditions could increase the probability of their occurring:

- Holding DBUTLTY LOCK OPTION=MOVER for a long time.
- Doing a dynamic extend of an area with a large new allocation.
- Having low priority jobs do maintenance to rows with lots of keys.
- Having a small LOGSPILL *d* Multi-User startup option percentage.
- Having a Log Area (LXX) with relatively few tracks.

How to Use

You can execute this command in either Single User or Multi-User mode. Execute the INIT function with the following command format:



Command

INIT

Invokes the function to format areas for CA Datacom/DB use.

Required Keywords

AREA=FXX

Identifies the area to be formatted as the Force Area.

,BLKSIZE=

Indicates the logical block size. The FXX is paired with the LXX in that it must have the same physical and logical block size.

Valid Entries:

An integer from 16384 to 98280

Default Value:

(No default)

,TASKS=

- Specifies the number of tasks to be supported.

Valid Entries:

1 through 9999

Default Value:

(No default)

Optional Keywords

,BLOCKS=*nnnnnn*

(Optional) Specifies the number of physical blocks per logical block. The FXX is paired with the LXX in that it must have the same physical and logical block size.

Valid Entries:

An integer from 1 to 3

Default Value:

1

Example JCL for Force Area

The following shows the command to initialize the Force Area with a physical block size of 16384 and TASKS=999.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL FXX,'force',9999,DA                        Force Data File
 // EXTENT SYSnnn,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
           INIT     AREA=FXX,BLKSIZE=16384,TASKS=999
 /*
 /&
 * $$ EOJ
```

Sample Report: INIT FXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             INIT AREA=FXX,BLKSIZE=16384,TASKS=999

                    FUNCTION=INIT
       ❷               AREA=FXX
                      BLKSIZE=16384
                      TASKS=999

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX




                    FXX - STATUS INFORMATION

                  { DEVICE TYPE -    3390
                  { BLOCK LENGTH - 16,384
                  { BLOCKS/TRACK -      3
                  {
             ❹    { BLOCKS IN USE -         3
                  { BLOCKS UNUSED -       147
                  { TOTAL BLOCKS -        150
                  { TOTAL TRACKS -         50
                  { PERCENT FULL -          2
```

❹

Status information about the Force Area at successful completion of the initialization.

Chapter 19: LINK (Link to the CXX)

When a data or Index Area is initialized, it is linked to the Directory (CXX) used in the INIT AREA=CXX function. This is done through the use of the CXX name. During all other processing when the data set is opened, an edit ensures that the CXX and the data set are linked together. If they are not, an error is produced.

The LINK function provides protection against JCL errors, such as pointing to the production database, not the test database. This is true for all area usage except the INIT. Therefore, users should use extra caution to ensure that the Directory in use and the data area or Index Area are correct when submitting an INIT request.

LINK

The LINK utility executes through the Multi-User Facility. It only executes when the Multi-User Facility is active.

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When to Use

Use the LINK function to link the Index Area and each data area in one database to a CXX without performing the INIT function. This would only be appropriate if you rename the CXX or copy a data area using a utility other than DBUTLTY to move data from one CXX to another.

If you used a utility other than DBUTLTY to backup the data area, it is still necessary to do the following:

1. First use DBUTLTY to INIT the IXX.
2. Next, null load the area.
3. Then use the non-DBUTLTY utility to restore the backup.
4. Follow the sequence above by doing the DBUTLTY LINK.

Note: Remember to DBUTLTY RETIX any other data areas in the data base to rebuild the IXX completely.

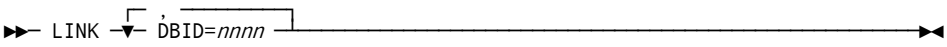
The LINK function opens tables in all areas of the specified database. Use this function only when all areas have been initialized and the data areas loaded.

The Accounting Facility tables must all be defined as closed during a Multi-User Facility enable before the LINK function can perform properly. After enabling the Multi-User Facility, you can perform the LINK function for the Accounting Facility PRM and data area database(s) and then open the desired tables.

You cannot use the LINK function to fix the Compound Boolean Selection Index Area. You must use the INIT function for the Compound Boolean Selection Index Area. See the *CA Datacom/DB Database and System Administration Guide* for more information.

How to Use

Multi-User mode is required to execute this command. Execute the LINK function with the following command format:



Command

LINK

Invokes the function to link the Index Area and each data area in one database to a Directory (CXX) without performing the INIT function.

Required Keywords

DBID=

Identifies the database or databases whose data and Index Areas are to be linked. DBID= may repeat. Each occurrence may specify either a single number or a range of numbers.

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database in a range not defined is ignored.

Valid Entries:

One or more valid database IDs or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250).

Default Value:

(No default)

Example JCL

The following shows the command to link the data and Index Areas in database 1 to the Directory that is used by the Multi-User Facility to which the job is connected.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          LINK     DBID=1
 /*
 /&
 * $$ EOJ
```

Sample Report: LINK

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                             General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             LINK DBID=1

                    FUNCTION=LINK
       ❷               DBID=1



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

      ❹             *                                   REQUEST COMPLETE
```

❹

Message indicating the maintenance completed successfully.

Chapter 20: LOAD (Build CXX or Data Area)

The LOAD function has two options. They are:

- Restore an entire Directory (CXX), or a Directory definition for a specific database.
- Build a data area and its associated Index Area (IXX) entries.

For information about DBUTLTY STATUS console command support for this function, see COMM STATUS (Print Status of Multi-User Jobs) (see page 180).

For information about the impact of table partitioning on the LOAD function, see The LOAD Utility (see page 41).

FILEIN may be used before a LOAD. See FILEIN (concatenate Input) (see page 335).

For information about dynamic extend considerations with regard to this function, see DBUTLTY Dynamic Extend Considerations, and for information about the DYNAMIC_EXTEND console-like command, see the *CA Datacom/DB Database and System Administration Guide*.

Overview

Use the LOAD function to perform the following:

- Restore an entire Directory (CXX) from a backup made using the BACKUP function.

  Caution If you are using SQL to access data, restoring the Directory from a current backup without restoring the DATA-DICT and Data Definition Directory (DDD) databases from backups taken at the same time can result in loss of synchronization which is critical for SQL operations.

  If you are using CA Ideal, restore the Virtual Library System files from a backup that was taken at the same time as the backup of the Directory to assure synchronization. See the CA Ideal documentation for details.

  To retain synchronization, when restoring the Directory, perform forward recovery. Beginning with Version 11.0, LOAD by DBID may require more GETVIS than was required in previous versions.

  If you cannot perform forward recovery, restore all data areas in the DATA-DICT database, except the HSD, and all data areas in the DDD database. Then, use the -HSD RESET transaction of the DDCFBLD utility to rebuild the HSD (see the *CA Datacom Datadictionary Batch Reference Guide*).

- Restore an individual database definition to the Directory.
- Load a data area with data for a single table that is in EXTRACT (user) format. See EXTRACT (Extract Data Table) (see page 321) for more information.
- Null load a data area. This causes all of the area's tables to be marked usable, but empty. It loads one table with no input data.
- Reorganize a data area into Native Key sequence by table.
- Restore a data area from a previous backup.

Before loading a data area for the first time, initialize the area using the INIT function. For a newly allocated database, both the Index Area (IXX) and all data areas must first have been initialized. After the primary initialization, you can load areas as often as required without performing an INIT.

LOAD CXX (Build CXX)

This option reads an input data set and restores the Directory data for all databases or for the database specified.

When to Use

Use this function to:

- Restore the Directory to a usable state or previous stable condition.
- Move the Directory to another device.
- Move the Directory to a larger extent.
- Restore a database definition from a backup to the same or a new DBID.

Restrictions

The following restrictions apply when attempting to build the Directory.

- No CA Datacom/DB system can be processing a Directory (CXX) when you request a full load to it, nor can processing occur on a database when loading that database.
- Do not load an Accounting database with the backup of an Accounting database from another system.
- You can load a CA Datacom/AD Directory only with a CA Datacom/AD Directory backup. (You can load a CA Datacom/DB Directory from a CA Datacom/DB Directory and a CA Datacom/AD Directory.)

SQL Processing

If you are using SQL to access data, restoring the Directory from a current backup without restoring the DATA-DICT and Data Definition Directory (DDD) databases from backups taken at the same time can result in loss of synchronization which is critical for SQL operations.

If you are using CA Ideal, restore the Virtual Library System files from a backup that was taken at the same time as the backup of the Directory to assure synchronization. See the CA Ideal documentation for details.

To retain synchronization, when restoring the Directory, perform forward recovery.

Full Directory Load

To perform a full load of the Directory, perform an INIT prior to the load. The INIT function can occur in a separate execution of DBUTLTY immediately preceding the LOAD or in a step preceding the step specifying the LOAD.

CA Datacom/DB restores all backup information during the LOAD with the following changes.

- CA Datacom/DB preserves the Directory name as established during the INIT. It ignores the name from the backup.
- CA Datacom/DB increments the run number from the backup by 10,000 and places that number in the initialized Directory. This ensures that the run unit number is not identical with that of the previous backup.

  If the 10,000 increment does not ensure that numbers do not repeat, back up the newly loaded Directory, initialize and reload the Directory to increment the run unit number by another 10,000.
- During the full load, CA Datacom/DB loads the information in the currently logical location in the Directory data set, not necessarily where it previously existed.

  Because of this movement, the Directory must not be open for processing. If the Directory is open for processing *and* CA Datacom/DB attempts to access the Directory, problems such as errors 13(126), 13(04), or 04(47) can occur. These errors do not cause the Multi-User Facility or CICS to abend, but you must close all CICS User Requirements Tables and end all batch jobs.
- The update and read-only user counts are zeroed during the LOAD.
- The BACKUP and LOAD sequence can be used to move the Directory or to increase the Directory size. The Directory (CXX) can be moved to a new device with the BACKUP and LOAD functions.

Directory Database Load

CA Datacom/DB updates the date/time of the maintenance each time the database is cataloged through the CA Datacom/DB. This catalog can be a direct catalog using CA Datacom/DB online or batch CATALOG or an indirect catalog using CA Datacom/DB DDCFBLD and the CXXMAINT utility function (option=DDPROD). CA Datacom/DB also updates the date/time of each DELETE. Therefore, create a new backup of the Directory after each maintenance. The CXXMAINT ALTER options change the Directory but are not considered maintenance. For additional information, see CXXMAINT (Maintain Directory) (see page 205).

Loading a single database definition with the same or a different database ID is possible. Ensure, however, that no maintenance has occurred since the backup being processed was made by comparing the maintenance dates with the current Directory. If these values do not match, CA Datacom/DB aborts the LOAD.

If you know that a backup is actually valid for all data areas even if the date/time is different, you can force the LOAD to leave the tables loaded. To do this:

1. Use CXXMAINT DELETE (Delete CXX Definitions) (see page 253) to delete the database,
2. Load the database (this load sets all the tables to *not loaded* status), then
3. Load the database again.

The last load in the procedure just given leaves the database information *as CA Datacom/DB found it on the backup.* That is to say, the last load does *not* provide a loaded status or initialized information if, at the time you performed the backup, the area was not then loaded.

CA Datacom/DB uses the database ID being loaded to find the desired database on the backup file. If a new database ID is not specified, CA Datacom/DB processes all information using the existing database ID.

If the database does not currently exist in the Directory, CA Datacom/DB adds the database and related definitions and information. However, CA Datacom/DB assumes the Index and data areas are not loaded since the database is new and the tables reflect that status. The Index and data areas are set as not initialized (unknown device type, zero blocks, zero tracks, and so on). Therefore, initialize the Index and data areas and load them from the backups, new input, or null load.

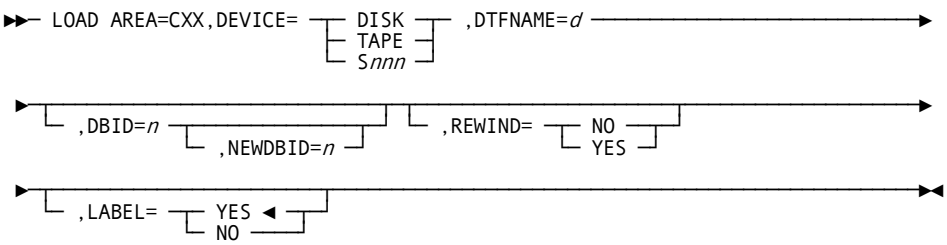
If the database currently exists in the Directory, CA Datacom/DB verifies that the update and read-only counts are zero and the date/time of the last maintenance is the same in the Directory and on the backup.

The load of one or more databases is always performed using the MUF, which must be enabled and the database closed.

How to Use

Multi-User mode is required to execute this command. Prior to version 12.0, either Single User mode or Multi-User mode could be used, but the LOAD function with AREA=CXX that uses a DBID is now executable in Multi-User mode only. This requirement prevents the execution of the function with the database open. It also ensures that the MUF is always in control and has knowledge of all information in the CXX to a CXX.

Execute the LOAD function using the following command format:



Command

LOAD

Invokes the function to build or restore an area.

Required Keywords

AREA=CXX

Identifies the area to be loaded as the Directory.

,DEVICE=

Specifies the device type for the input data set.

If DEVICE=TAPE is specified, the system logical unit (SYS) number must be 5 (SYS005).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

,DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the input data set. If DEVICE=TAPE, this name must match the name which you code in your z/VSE JCL TLBL statement. If not, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

Optional Keywords

,DBID=

Identifies the database selected for loading a specific database to the Directory.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), specifies whether tape labels are processed. We recommend that you always use labeled tapes.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,NEWDBID=

Specifies a new DBID for the data to be loaded.

Note: Do not use this parameter for databases accessed by SQL or CA Dataquery. These products must have the correct DBID to reference additional information from CA Datacom/DB about the production database. If the NEWDBID= parameter is used for a database accessed by CA Ideal, the corresponding ASSIGN DBID or ALTER DBID command must be run before running the application. See CA Ideal documentation for details.

Valid Entries:

Valid DATACOM-ID for a database

Default Value:

(No default)

REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates that the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

Example JCL (Input from Tape)

The following shows the command to load the Directory backup into the current Directory. The input data set is CXXIN.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',9999,DA                        Directory data set
 // EXTENT SYSnnn,volser,1,0 rel trk/blk,number trks/blks
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL CXXIN,'cxx.bkup'                         Directory Backup
 // EXEC DBUTLTY,SIZE=750K
        LOAD  AREA=CXX,DEVICE=TAPE,DTFNAME=CXXIN
 /*
 /&
 * $$ EOJ
```

Example JCL (Input from Disk)

The following shows the command to load the Directory backup into the current Directory. The input data set is CXXIN.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',9999,DA                          Directory data set
 // EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
 // ASSGN SYS005,DISK,VOL=volser,SHR
 // DLBL CXXIN,'cxx.bkup'                           Directory Backup
 // EXTENT SYS005,volser
 // EXEC DBUTLTY,SIZE=750K
         LOAD  AREA=CXX,DEVICE=DISK,DTFNAME=CXXIN
 /*
 /&
 * $$ EOJ
```

Sample Report: LOAD CXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             LOAD  AREA=CXX,DEVICE=TAPE,DTFNAME=CXXIN

                    FUNCTION=LOAD
       ❷               AREA=CXX
                      DEVICE=TAPE
                      DTFNAME=CXXIN


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

       ❹            *                                    REQUEST COMPLETE
```

❹

Message indicating the LOAD function completed successfully.

LOAD (Build a Data Area)

This option reads an input data set and writes the data into a CA Datacom/DB data area. The Index Area (IXX) entries are updated to reflect the data being loaded. This function prepares the area for any subsequent data access request made through CA Datacom/DB.

When to Use

Use the LOAD Data Area option to:

- Restore a data area from a previous backup.

  Caution If you are using SQL to access data, restoring the areas of the DATA-DICT and Data Definition Directory (DDD) databases from current backups without restoring the Directory from a backup taken at the same time can result in loss of synchronization which is critical for SQL operations.

  If you are using CA Ideal, restore the Virtual Library System files from a backup that was taken at the same time as the backup of the Directory to assure synchronization. See the CA Ideal documentation for details.

  To retain synchronization, when restoring the Directory, perform forward recovery.

- Convert to a Unique Row Identifier (URI) format.

- Load a data area with data for a single table that is in EXTRACT (user) format. See EXTRACT (Extract Data Table) (see page 321) for more information.
- Null load a data area (cause all of the tables for this area to be marked usable, but empty) by loading one table with no input data.
- Reorganize a data area into Native Key sequence by table.

Important! Do not back up an Accounting Facility PRM table from one system and load it to another system.

Loading an Existing Area

LOAD always loads an entire area using whatever data you provide as input (if any). Any data that exists in the area when a LOAD is executed is completely overwritten by the utility.

Loading a New Area

Before a data area can be loaded, you must have initialized the data area once using the INIT function. You must also initialize any newly allocated Index Area (IXX). Do not reinitialize the Index Area when loading subsequent data areas in the database or previously created Index entries for the other areas are lost. When reloading an entire database, initializing the Index Area before performing the first data area LOAD generally speeds processing. A data area or Index Area need only be initialized once.

Loading a Database

When loading a database, take note of the following rules:

- The SORT= option is required.
- FORMAT must be BACKUP or NONE.
- If BACKUP input is used, only one input file is supported.
- An INIT of the index must precede the LOAD, but not necessarily in the same step.
- All areas must be loaded as URI. The URI= keyword may not be set to NO.

Null Loading

If no data is required in any of the tables in the area prior to application task processing, all tables in the area can be prepared by performing a null load of the area (specify the parameter FORMAT=NONE). A null load is also useful to clear all tables in a data area. For example, you might design an application to store transactions for processing in tables in a data area.

Once all transactions have been processed, a null load of the area is more efficient to clear all transactions than deleting each of the individual transactions. As part of null loading a data area, CA Datacom/DB must search the Index for entries pointing to previous data in the area and delete those entries. This deletion occurs more rapidly and completely if the SORT keyword value is 1.

Loading Special Rules with Database having a Multi-Dataset Index

A database with Multi-Dataset Index areas has special requirements including the following:

- SORT must be specified as greater than 0 (or FORMAT=NONE).
- URI must be set to YES or allowed to default.
- The database can have no table with RECOVERY NO as part of its definition.

Dynamic Extend during LOAD

Dynamic extend of an index area or data area is attempted if requirements are met. If requirements are not met, dynamic extend is not attempted. The requirements are:

- SORT must be specified as greater than 0 or FORMAT=NONE.
- The URI option must be set to YES or allowed to default.
- No table in the database can have a table with RECOVERY NO specified in its definition.

Loading from a Backup

If you are loading data from a backup, the LOAD function *only* restores the data if the table IDs are identical when LOADID=YES (the default). LOADID=NO can be used to match on table names, but LOADID=NO cannot be specified if either URI=NO is specified or if any table in the area was not defined to CA Datacom/DB with RECOVERY=Y. REMAP= can be used to force matching on different table names. This feature is available only for areas loaded as URI.

Loading areas that have been dynamically or manually extended require that the data set size reflect all tracks that previously existed. Load attempts made to an area not reflecting all extended tracks receive a return code 76 (009).

If you change the size of a row in an area that uses compression and use a backup taken prior to the change, the load fails if you do not specify the keyword MISMATCH=IGNORE. With MISMATCH=IGNORE specified, longer rows are truncated to the table row size in the Directory (CXX) and rows that are shorter are padded with blanks (hex 40s). Without MISMATCH=IGNORE specified, you receive message

DB13001E - UNEXPECTED RETURN CODE 21 (74)

when rows are longer than the CXX size, and message

DB13008E - INPUT RECORD LENGTH nnn DOES NOT MATCH THE TABLE ttt RECORD LENGTH nnn

when rows are shorter than the CXX size.

Reorganizing Data Areas

LOAD can also be used to reorganize the records in a data area into Native Key sequence. First obtain a backup using BACKUP SEQ=NATIVE, then provide this backup to the LOAD function. The data is reloaded in Native Key sequence.

Loading to Perform RECOVERY

LOAD is the first step in performing a recovery after a data area has become damaged. If you intend to perform forward recovery to completely bring the data up to date, the input to LOAD must generally be a backup that was created using SEQ=PHYSICAL and RECID=YES. See Back Up Data Area (see page 95) and RECOVERY (Rebuild a Database) (see page 443) in this manual and see the *CA Datacom/DB Database and System Administration* Guide for further details about forward recovery.

Sorting to Optimize Index Processing

The SORT option is highly recommended because it allows the Index processing to execute more efficiently by waiting until the end of the load to establish reference points in the Index. This usually improves the load time significantly. If you use multiple load statements in a single step, place the largest sort value first.

Converting to URI

Before converting a data area to the Unique Row Identifier (URI) format, ensure that it is stable. Recovery Files made prior to the conversion cannot be used after the conversion.

When converting to URI, the backup supplied to the DBUTLTY LOAD must have been created with RECID=NO specified.

In a data area using URI, all tables must be defined in CA Datacom/DB with the attribute RECOVERY=Y. When converting a data area to URI, if DBUTLTY LOAD encounters a table defined with RECOVERY=N, it loads the data area but does not perform the conversion to URI and issues the message: DB13040W - TABLE aaa LACKS RECOVERY INFO: AREA bbb PROCESS AS NON-URI.

Loading from a Sequential File

You can load a single table from preexisting data that contains the expanded image of the records. Such data can be created as a sequential file from an application program or can be created using the EXTRACT function of DBUTLTY, EXTRACT (Extract Data Table) (see page 321). Indicate either form of input by using the FORMAT=EXTRACT parameter.

VSAM Input

CA Datacom/DB accepts VSAM input during the LOAD process.

User Compression

The LOAD function now can read either expanded or compressed input data and load it based upon the compression option specified in the Directory (CXX). If a RECID=YES backup is used to load compressed rows as uncompressed, an error is generated if any row on the backup cannot be placed into the same exact block from which it was backed up.

Loading Tables with Integrity Constraints

The LOAD **does not** verify any of the constraints at load time. It marks tables which may contain constraint violations as CHECK PENDING, a status which is reported on a Directory report. CONFIRM (Verify Constraint Integrity) (see page 199) provides information on how to verify integrity constraints and remove the CHECK PENDING status.

Note: You must CONFIRM tables in the following order, from referenced-to-referencing, or parent-to-child, that is to say, the referenced table cannot be in check pending status. For example, if there is a foreign key from a REMARKS table to a LINE_ITEM table on PO_Nbr and Line_Item_Nbr, and the LINE_ITEM table has a foreign key that references the PO table on PO_Nbr, the PO table must be loaded first, then the LINE_ITEM table can be confirmed, then the REMARKS table can be confirmed. Therefore, the confirm order would be: from PO table to LINE_ITEM table to REMARKS table.

Data Validation

If the fields in a table being loaded are defined through SQL, or through CA Datacom/DB with DBEDITS=Y, the data in the fields is validated. Fields defined as the following data types are checked:

- Decimal
- Float
- Variable
- Binary specified as SQL-DATE, SQL-TIME, or SQL-STMP

If invalid data is found for a field, the record is hex dumped and the load fails. There is no override to load invalid data. To load the table, you must correct the data or change the FIELD occurrence definition to DBEDITS=N.

Fields which have MIXED semantic type with DBEDITS=Y attribute-value are edited to ensure that every Shift-Out character is matched to a Shift-In character an even number of bytes away. This editing takes place only if the Directory is set for DBCS support mode with the DBUTLTY CXXMAINT function CXXMAINT ALTER DBCS (Activating DBCS Support) (see page 209).

Force Default Fields

Fields which are defined through CA Datacom/DB with FORCEADD=Y or FORCEUPD=Y are not modified by the LOAD function.

Loading Multiple Tables Into a Data Area

The following are the options for initially loading multiple tables into an area.

LOAD/BACKUP Method

1. Load the first table using the LOAD function and back it up using the BACKUP function. Specify SEQ=NATIVE or SEQ=PHYSICAL. If SEQ=PHYSICAL is specified, RECID=NO must be entered. Load the next table and back it up. Continue to use the LOAD/BACKUP functions for each table you are loading. If the area is clustered, specify CLUSTER=DENSE for each load.
2. After completing the last BACKUP, back up each table as a single file via a system utility, such as DITTO, SORT/MERGE, and so on. Use this file to load all the tables together.
3. If any of these tables share a Native Key ID or if the area is clustered, then you must do one additional backup with SEQ=NATIVE, and then load it to cluster the tables. If the area is clustered, specify CLUSTER=SPARSE.

LOAD/MASSADD Method

1. Load the first table in the area using the LOAD function.
2. Then use the MASSADD function to load the other tables in the area. See MASSADD (Add Records to Table) (see page 427) for more details. Using this combination eliminates having to back up each table individually and having to combine the tables to create a single input source.

DBFLSUB Method

To initially load a multiple table area in fewer steps, create an application program that invokes the DBFLSUB subroutine to load multiple tables merged on Native Key IDs and then on Native Key values. See the *CA Datacom/DB Database and System Administration Guide* for details on using the DBFLSUB subroutine.

Adding a Table to a Data Area

To add a table to an existing area which already contains one or more tables:

1. Create a backup of the area using the BACKUP function. Specify SEQ=NATIVE or SEQ=PHYSICAL. If SEQ=PHYSICAL is specified, RECID=NO must be entered.
2. Load the new table into the area with LOAD.

3. Back up the area using BACKUP. Specify SEQ=NATIVE or SEQ=PHYSICAL. If SEQ=PHYSICAL is specified, RECID=NO must be entered.
4. Back up the data as a single file via a system utility, such as DITTO, SORT/MERGE, and so on. Use this file to load all the tables together.
5. If any of these tables share a Native Key ID or if the area is clustered, then you must do one additional backup with SEQ=NATIVE, and then load it to cluster the tables. If the area is clustered, specify CLUSTER=SPARSE.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any Multi-User Facility for read.
- The database may *not* have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO*

- The database will be opened for update.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when MULTUSE=YES*

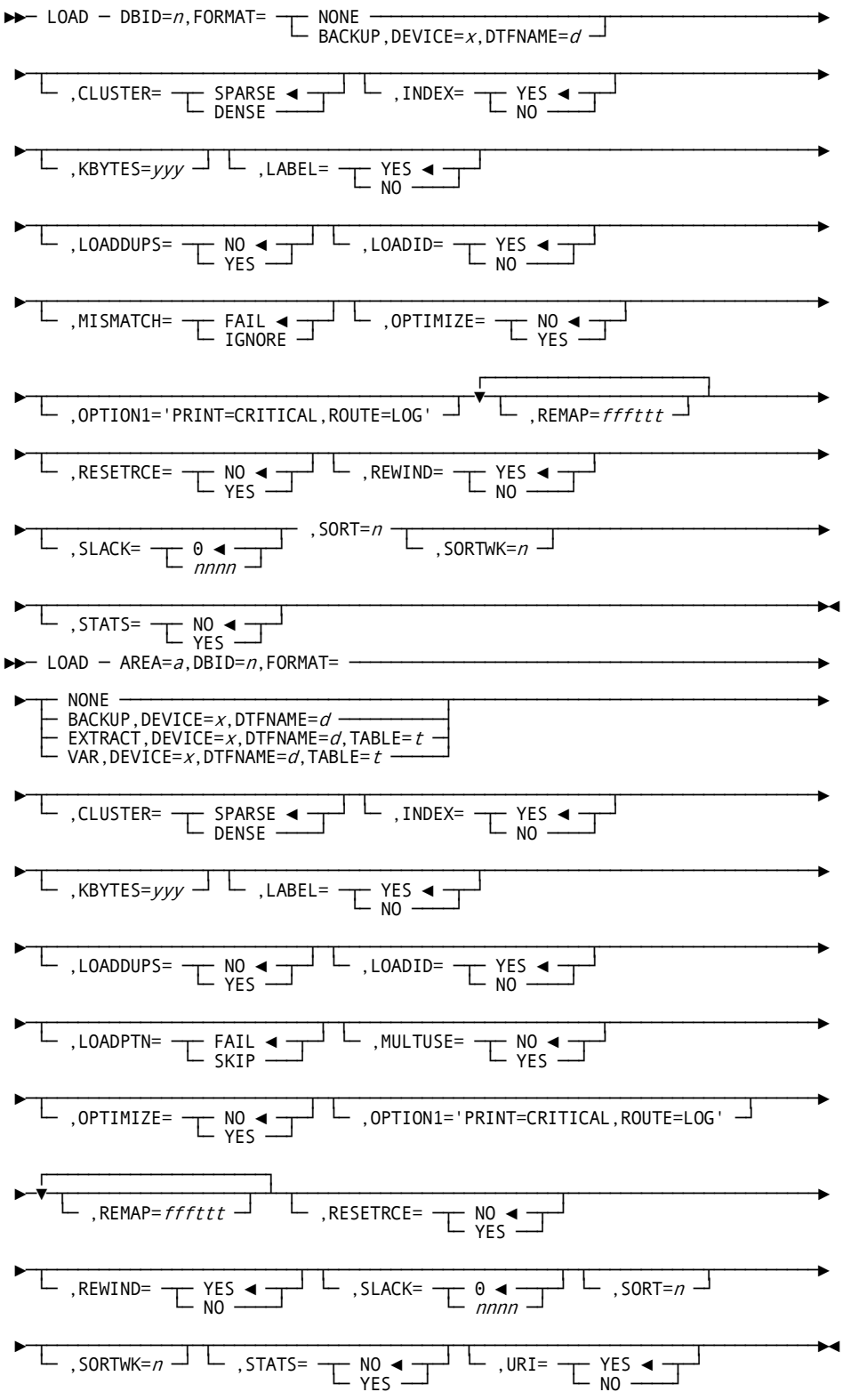
- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY
- The area may have no table in an open URT
- The area may *not* have any other MULTUSE=YES utility executing
- The area must be closed

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets LOAD executing this Multi-User Facility this database or area.

How to Use

You can execute this command in either Single User or Multi-User mode. Execute the LOAD function using the following command format.


LOAD – DBID=n,FORMAT= NONE
BACKUP,DEVICE=x,DTFNAME=d
,CLUSTER= SPARSE ◄ DENSE
,INDEX= YES ◄ NO
,KBYTES=yyy
,LABEL= YES ◄ NO
,LOADDUPS= NO ◄ YES
,LOADID= YES ◄ NO
,MISMATCH= FAIL ◄ IGNORE
,OPTIMIZE= NO ◄ YES
,OPTION1='PRINT=CRITICAL,ROUTE=LOG'
,REMAP=ffffff
,RESETRCE= NO ◄ YES
,REWIND= YES ◄ NO
,SLACK= 0 ◄ nnnn
,SORT=n
,SORTWK=n
,STATS= NO ◄ YES
LOAD – AREA=a,DBID=n,FORMAT=
NONE
BACKUP,DEVICE=x,DTFNAME=d
EXTRACT,DEVICE=x,DTFNAME=d,TABLE=t
VAR,DEVICE=x,DTFNAME=d,TABLE=t
,CLUSTER= SPARSE ◄ DENSE
,INDEX= YES ◄ NO
,KBYTES=yyy
,LABEL= YES ◄ NO
,LOADDUPS= NO ◄ YES
,LOADID= YES ◄ NO
,LOADPTN= FAIL ◄ SKIP
,MULTUSE= NO ◄ YES
,OPTIMIZE= NO ◄ YES
,OPTION1='PRINT=CRITICAL,ROUTE=LOG'
,REMAP=ffffff
,RESETRCE= NO ◄ YES
,REWIND= YES ◄ NO
,SLACK= 0 ◄ nnnn
,SORT=n
,SORTWK=n
,STATS= NO ◄ YES
,URI= YES ◄ NO

Command

LOAD

Invokes the function to build or restore an area.

Required Keywords

DBID=

Identifies the database containing the area to be loaded. Beginning with Version 11.0, LOAD by DBID may require more GETVIS than was required in previous versions.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,DEVICE=

Specifies the device type for the input data set.

This parameter is not allowed if you specify FORMAT=NONE. This parameter is required when you specify any value other than NONE for FORMAT=.

DEVICE=VSAM specifies a VSAM cluster, accessed either directly or through a VSAM path. The other values specify physical sequential data sets.

When DEVICE=TAPE is specified, the system logical unit (SYS) number must be 5 (SYS005).

Use DEVICE=S*nnn* to specify a specific SYS number (for tape only).

Valid Entries:

DISK, VSAM, TAPE, or S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

,DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the input data set.

This parameter is not allowed if you specify FORMAT=NONE. This parameter is required when you specify any value other than NONE for FORMAT=.

If DEVICE=TAPE, this name must match the name which you code in your z/VSE JCL TLBL statement. Otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

,FORMAT=

Specifies the format of the input data.

BACKUP

Indicates the data being read was created by using the BACKUP function of DBUTLTY.

EXTRACT

Specifies that the data being loaded is expanded fixed-length images of the data records. Such data can have been created by an application program or when the EXTRACT function of DBUTLTY is used. This option requires the TABLE parameter to direct the data to the correct table.

Note: The block size of the input file cannot exceed 32K - 1.

NONE

Specifies a null area load, no data being loaded. If the current index has entries for more than several hundred records, using SORT= with a value of 1 is recommended. As part of null loading a data area, CA Datacom/DB must search the index for entries pointing to previous data in the area and delete those entries. This deletion occurs more rapidly and completely if SORT=1.

Note: If you specify FORMAT=NONE, do not specify the DEVICE=, DTFNAME=, and TABLE= parameters.

VAR

Indicates that the input data has the IBM variable-blocked format. This option is used to load variable-length data to be processed by the CA Datacom VSAM Transparency and must be specified if the input is a VSAM file with variable-length records and you want to load directly to CA Datacom/DB. The table must be defined with user compression. See the *CA Datacom VSAM Transparency User Guide* for more information.

The VAR option requires the TABLE parameter to direct the data to the correct table. The VAR option is needed only on the first load of an area with input that is IBM variable-blocked format. Thereafter, the BACKUP function produces a format that can be input to the LOAD using the FORMAT=BACKUP option.

Valid Entries: BACKUP, EXTRACT, NONE, and VAR

Default Value: (No default)

Optional Keywords

AREA=

Identifies the area to be loaded. If you omit the AREA= keyword and use LOAD DBID=, you can do a load of a full database. For additional comments relating to a database level load, see the description of the DBID= keyword.

Valid Entries:

DATACOM-NAME of the area in the database specified in the DBID= parameter

Default Value:

(No default)

BLKSIZE=

We do not recommend that you specify a BLKSIZE=, because this parameter is ignored. It is provided for compatibility with an earlier version of CA Datacom/DB, because of the possibility that it may be part of previously specified JCL. If BLKSIZE is specified, it must be a number.

,CLUSTER=

Specifies whether the data is to be loaded sparsely with each cluster to its own data block(s) or densely with data packed in the next block location with no regard to cluster value.

DENSE

Should only be used during the initial loads of a multi-table area to build the required format and sequence of the input records to load the clustered area correctly.

SPARSE

Is required for normal clustering.

Valid Entries:

DENSE or SPARSE

Default Value:

SPARSE

,INDEX=

Specifies whether the data indexes are to be built. Specifying NO is only intended for use with forward recovery in cases where the total recovery time is expected to be better by doing no data index work until a RETIX function is issued after the recovery. The index must be and is built with correct URI and space information.

Valid Entries:

NO or YES

Default Value:

YES

,KBYTES=*nnnn*

Specifies the number of 1024-byte blocks of memory needed to build the buffers for CA Datacom/DB. CA Datacom/DB chain writes up to one cylinder of data buffers if sufficient memory is available. In general, providing buffers to contain 5-6 tracks of data is optimal. You can use the following formula to calculate the value for this parameter:

blksize * #-of-blks-per-trk * #-trks * 2 / 1024

Where:

blksize

The data area block size. See the Directory (CXX) report for this value.

#-of-blks-per-trk

Number of blocks that fits on a track for the specified device type. You can see the Directory (CXX) report for this value.

#-trks

Number of tracks that the user desires to have written for each physical I/O. This need not exceed the number of tracks per cylinder.

2

For double buffering.

Example:

A 4K block size on a 3390 device using two track writes:

(4096 * 12 * 2 * 2 / 1024) = 192

Note: The value obtained should be rounded up to the next whole number. The maximum value cannot exceed 9999. Sufficient memory is required for these extra data buffers.

Valid Entries:

1 to 9999 (integers only) multiplied by 1024 is the number of bytes requested through GETMAIN.

This number multiplied by 1024 is the number of bytes allocated for data buffer chaining in the partition. If loading by database, the number of bytes allocated is: (KBYTES*1024)*(areas in the database)). The LOAD does not run if sufficient memory is not available.

Default Value:

four tracks of memory buffering

Note: If a value is specified which results in less than 4 tracks of data, the input is rounded up during execution to be 4 tracks.

LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), specifies whether tape labels are processed. We recommend that you always use labeled tapes.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,LOADDUPS=

Specifies whether to load duplicate Master or Unique Key values when the table is defined as having no duplicates. If CA Datacom/DB encounters a duplicate value for a Master Key for a table defined with no duplicate Master Key or for a Unique Key, CA Datacom/DB issues a return code 10 and aborts the run. If you specify YES for this parameter, CA Datacom/DB accepts your override and loads the key values. CA Datacom/DB produces a snap dump of the first 500 duplications. When MULTUSE=YES, the snap dumps are written to the Multi-User Facility Statistics and Diagnostics Area (PXX). See the section on using dumps in the *CA Datacom/DB Database and System Administration Guide* for a sample.

Valid Entries:

NO or YES

Default Value:

NO

,LOADID=

(Applies to FORMAT=BACKUP) Specifies match on input row to a table in the area or database to be loaded using the table DATACOM-ID (YES) or the table DATACOM-NAME (NO). For a normal backup and load, this parameter is not important because the input row matches on both. However, when changing definitions between the backup and load, it is critical to a successful load. If changing table names or IDs, the matching allows flexibility. It may be necessary when backing up from one Directory (CXX) and loading into another.

If the input or output is a record in a partitioned table, the parameter must be specified as LOADID=NO to force matching on name. The name to be matched is only the Full Parent name or an unpartitioned table name.

If the table is a partition, a row can be rejected and not loaded if the partition value criteria does not match. How the utility handles the rejection is determined by the value set for the LOADPTN= parameter.

For more information about table partitioning and the LOAD function, see The LOAD Utility (see page 41).

Valid Entries:

NO or YES

Default Value:

YES

,LOADPTN=

Specifies how the utility handles a rejected row for a table partition. If it is set to SKIP, the row is ignored with no error condition. If it is set to FAIL, the row is dumped, and the utility function terminates.

Valid Entries:

FAIL or SKIP

Default Value:

FAIL

,MISMATCH=

Specifies whether to load records when the length of the records on the backup do not match the record size in the Directory (CXX).

We strongly recommend that a backup be taken of the area to be loaded before using LOAD with MISMATCH=IGNORE, and that you exercise caution when executing a LOAD with MISMATCH=IGNORE. If an expanded row length exceeds the CXX table length and MISMATCH=IGNORE is specified, all data beyond the CXX length is truncated and therefore lost. If an expanded row length is less than the CXX table length and MISMATCH=IGNORE is specified, the balance of the expanded row is filled with spaces (hex 40s).

Valid Entries:

FAIL or IGNORE

Default Value:

FAIL

,MULTUSE=

(For area level DBUTLTY control only.) Specifying MULTUSE=YES allows a single area in a database to be loaded with no loss of read and update access to other areas in the database. When MULTUSE=YES, the function executes in a new format that is a combination of Multi-User Facility and Single User modes. The database involved must be able to be opened for update in the Multi-User Facility address space. The area must have no tables open. If the database is currently open, CA Datacom/DB closes this area with the area close (you would need to close the area yourself *only if* the database is open and you need Multi-User Facility to close it, so that any type of open data set lock can be released, which could be necessary in order to scratch and reallocate the data set). Once started, the function does not allow other tasks to open any tables in the area being processed. Other tables in the involved database, but not in the area, may be open for read or update in the same Multi-User Facility. With MULTUSE=YES:

- Area is first opened in Multi-User Facility for the LOAD.
- Area is also opened in Single User mode.

- Data area processing occurs in the utility address space.
- Key build/sort occurs in the utility address space.
- Index load/merge occurs in Multi-User Facility address space. Packages of index entries (about 32k bytes) are sent from utility to Multi-User Facility.

When MULTUSE=YES, the SORT= option must be specified but not as zero (0). The data area must currently be set to the URI format. This can be done with a data INIT or a previous LOAD using URI=YES. The keyword URI=NO may not be specified (the area may only be loaded in the URI format).

When MULTUSE=NO is specified or is allowed to default in the LOAD and RETIX functions, the function executes as in versions prior to Version 11.0, that is, the database involved may not be open for update, and all processing executes in the Single User mode.

For more information on area level control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

NO or YES

Default Value:

NO

,OPTIMIZE=

Specifies whether the Index entry creation optimization option is to be used.

YES

All Index entries are not to be completed until the load terminates. This option enhances performance by deferring Index pointer validation and complete Index structure update.

Note: OPTIMIZE=YES is ignored if MULTUSE=YES is also specified.

Note: When YES is selected, if this load fails for any reason, reinitialize the Index Area. If any other data areas are already loaded, re-create the Index entries using the RETIX function of DBUTLTY, or reload all areas. If you select YES, you must also specify a value in the SORT= parameter.

NO

All Index entries are to be validated as they are created.

Valid Entries:

NO or YES

Default Value:

NO

,OPTION1=

(*z/VSE only*) Specifies the options for the associated sort/merge program application. The values for this keyword must be 44 characters or less and they must be enclosed in quotes. Depending on the features of your installation's sort, the values that apply to this keyword can vary. CA Datacom/DB does not edit anything that you place inside of the quotes.

Valid Entries:

Any values that your installation's sort allows. These values must be less than 44 characters and be inside quotes.

Default Value:

CRITICAL for PRINT

LOG for ROUTE

,REMAP=*fffttt*

(Applies to FORMAT=BACKUP, values are ignored by other formats.) When LOADID=NO is specified for matching input data to output tables based upon table name, REMAP= allows a table name to be remapped. This parameter allows you to load data when the table has been renamed between the backup and the load.

To use REMAP=, specify six characters *fffttt* after REMAP=, where *fff* is the input table name (the *from* table on input) and *ttt* is the output or current Directory (CXX) name (the *to* table in the database ID that is being loaded). CA Datacom takes the data records for the *from* table and loads them into the *to* table, effectively renaming the table.

Message DB13022E can occur if the BACKUP used RECID=YES.

Valid Entries:

six characters *fffttt* as previously described.

Default Value:

(No default)

,RESETRCE=

(Applies to FORMAT=BACKUP, values are ignored by other formats.) When a table is defined with recovery, the Record Control Element (RCE) contains the Transaction Sequence Number (TSN) of the transaction that last did maintenance to a record. CA Datacom/DB uses this information to know if secondary exclusive control is necessary on a record. The master transaction sequence number is stored and maintained in the Log Area. If you have multiple Log Areas that share in any context the updating of a data area (though obviously not at the same time), the master transaction sequence number must be in a different range or unnecessary waits may occur. Using RESETRCE can prevent this from happening.

For example, if you have production and test systems using similar numbers through the Log Area, you could use RESETRCE to ensure that no waiting is done on the wrong transaction. Such as, if you had backed up production data, loaded it to the test system, and initialized the Log Area on a system, the transaction sequence numbers would be repeated on one system, resulting in possible waits on the wrong transaction. A backup and load with RESETRCE would prevent this condition, avoiding unnecessary waits.

Note: For EXTRACT input or when RESETRCE=YES is specified, the TSN is set to zero. Otherwise, the TSN is copied from the BACKUP record.

Valid Entries:

NO or YES

Default Value:

NO

REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates that the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

,SLACK=

Specifies the number of bytes to reserve in each data block for future expansion of records in that block, or for addition of records when using a space reclamation option. The number entered applies only to blocks loaded to an area during this execution of the LOAD function. Blocks not loaded with data use the SLACK value specified in the AREA definition in CA Datacom/DB.

Valid Entries:

A number that is less than the block size, up to a maximum of 9999.

Default Value:

0

,SORT=

Activates the Index entry sort option.

Note: The sort option is required when specifying the LOAD function for the full database, or if MULTUSE=YES.

When you specify a value greater than zero, no Index entries are created as the data is loaded. The required Index entries are sorted at the end of the data LOAD and are merged into the Index in ascending sequence. Selecting this option enhances the performance of the LOAD process for large databases.

The SORT option is highly recommended when the number of records exceeds several hundred. Specify that the sort is to be used by entering a non-zero number. If sort is selected by entering a non-zero number, the Index entries in the Index Area (IXX) are not deleted by a separate Index pass. The Index Area is updated and only the Index blocks that reflect change are written.

If the SORT option is used, the partition size should be at least nine megabytes (9MB).

Note: Do *not* specify over 99 million unless you verify that the sort package you are using allows a larger number.

Specifying SORT=0 (or any number of zeros) is the same as not specifying the keyword.

Many features introduced in previous releases require a non-zero value for SORT=, and the following two features are also blocked during LOAD or RETIX executions, when the sort option is not being used are, as follows:

- on-demand dynamic extend of any data area or Index Area
- Execution against a database with Multi-Dataset Indexes

Most user-JCL is expected to already have a non-zero value for SORT= specified. If, however, a user has a LOAD or RETIX function coded in their JCL with SORT=0 specified or defaulted (SORT= not specified at all), this information is added to the history report, and the first time in each MUF execution that an area is opened in a database that has a LOAD or RETIX coded without the sort option, an informational message is generated. The message alerts you to the condition and can therefore help you know to first add the sort option, then see the results.

Note: A load with FORMAT=NONE (and either SORT=0 or SORT= not specified) is handled by DBUTLTY in a special, "simulated sort" manner and therefore does not require SORT= to be specified as a number greater than 0. For consistency, however, we recommended that SORT=1 be specified with FORMAT=NONE.

Two messages related to using LOAD and RETIX without the sort option are messages DB02809I and DB13128E. For details about these messages, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

0 to 99999999999

Default Value:

(No default)

,SORTWK=

Specifies the number of sort work areas.

Note: Do not specify a 2-digit number unless the sort package you are using allows a 2-digit number. Sort packages that do not support a 2-digit number can fail in an unpredictable manner, if you specify a 2-digit number.

Valid Entries:

0 to 99

Note: Do not use two-digit numbers unless the SORT package to be used supports the larger number.

Default Value:

3

,STATS=

Specifies whether compression statistics are to be computed and displayed.

YES specifies to compute and print the compression statistics. Specifying YES when no tables are compressed significantly increases LOAD runtime. Only use YES to determine the space savings of compression.

Valid Entries:

NO or YES

Default Value:

NO

,TABLE=

Indicates a single table is to be loaded. This parameter is required when EXTRACT or VAR is specified for FORMAT. It cannot be specified when FORMAT=BACKUP or FORMAT=NONE.

Valid Entries:

DATACOM-NAME of a table in the database and area being loaded

Default Value:

(No default)

,URI=

Specifies whether to use URI record ID formats. The URI format is required for SQL access and for tables with unique keys. All tables in the area must be defined to CA Datacom/DB with RECOVERY=Y.

If you have defined the tables in this area with unique keys, you must specify URI=YES.

When the backup being loaded was created with RECID=YES, DBUTLTY ignores this keyword and loads the data area using the record ID format of the backup.

Note: URI=NO may not be specified if MULTUSE=YES is specified. Allowing the code to run in the URI mode is highly recommended.

For further information about URI, see the *CA Datacom/DB Database and System Administration Guide*.

Valid Entries:

NO or YES

Default Value:

For RECID=YES backups: format of the backup

Otherwise, Default Value:

YES

Example JCL (Input from Tape)

The following shows the command to load the DEM area in database 001 (writing to DEM001 data set), to estimate the number of keys to sort at 500, to allow no bytes of slack for record growth, to compute no compression stats, and to use the default buffer area. The data being read was created by a backup of the area DEM on tape.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',9999,DA           Index for DBID 1
 // EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
 // DLBL DEM001,'db001.dem',9999,DA             Area DEM in DBID 1
 // EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL BACKDEM,'input'                        Input data set
 // ASSGN SYS001,device type,VOL=volser,SHR
 // ASSGN SYS002,device type,VOL=volser,SHR
 // ASSGN SYS003,device type,VOL=volser,SHR
 // DLBL SORTWK1
 // EXTENT SYS001,volser,,,rel trk/blk,number trks/blks
 // DLBL SORTWK2
 // EXTENT SYS002,volser,,,rel trk/blk,number trks/blks
 // DLBL SORTWK3
 // EXTENT SYS003,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
        LOAD  AREA=DEM,DBID=1,FORMAT=BACKUP,
              CLUSTER=DENSE,DEVICE=TAPE,DTFNAME=BACKDEM,
              LOADDUPS=NO,OPTIMIZE=YES,SORT=500
 /*
 /&
 * $$ EOJ
```

Example JCL (Input from Disk)

The following shows the command to load the DEM area in database 001 (writing to DEM001 data set), to estimate the number of keys to sort at 500, to allow no bytes of slack for record growth, to compute no compression stats, to print sort messages on SYSLST, and to use the default buffer area. The data being read was created by a backup of the area DEM on disk.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',9999,DA           Index for DBID 1
 // EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
 // DLBL DEM001,'db001.dem',9999,DA             Area DEM in DBID 1
 // EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL BACKDEM,'input'                        Input data set
 // EXTENT ,volser
 // ASSGN SYS001,device type,VOL=volser,SHR
 // ASSGN SYS002,device type,VOL=volser,SHR
 // ASSGN SYS003,device type,VOL=volser,SHR
 // DLBL SORTWK1
 // EXTENT SYS001,volser,,,rel trk/blk,number trks/blks
 // DLBL SORTWK2
 // EXTENT SYS002,volser,,,rel trk/blk,number trks/blks
 // DLBL SORTWK3
 // EXTENT SYS003,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
        LOAD  AREA=DEM,DBID=1,FORMAT=BACKUP,
              DEVICE=DISK, DTFNAME=BACKDEM,
              OPTION1='PRINT=ALL,ROUTE=LST',SORT=500
 /*
 /&
 * $$ EOJ
```

Sample Report: LOAD Data Area

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             LOAD   AREA=DEM,DBID=1,FORMAT=BACKUP,CLUSTER=DENSE,
                              DEVICE=TAPE,DTFNAME=BACKDEM,LOADDUPS=NO,OPTIMIZE=YES,SORT=500

                    FUNCTION=LOAD
       ❷               AREA=DEM
                       CLUSTER=DENSE
                       DBID=1
                       DEVICE=TAPE
                       DTFNAME=BACKDEM
                       FORMAT=BACKUP
                       LOADDUPS=NO
                       OPTIMIZE=YES
                       SORT=00000500

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX


 ❹

BACKUP FILE INFORMATION

JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA

DBDVL      56,377   5/25/2003 16.26.35  PRODCXX       1  PHY   NO    DEM



INPUT FILE INFORMATION          OUTPUT TABLE INFORMATION

AREA  TABLE  ID     RECORDS     AREA  TABLE  ID     RECORDS     NOTES

DEM   PNC     6           5     DEM   PNC     6           5
DEM   PNM     5           3     DEM   PNM     5           3
DEM   POH     3           5     DEM   POH     3           5
DEM   POL     4           4     DEM   POL     4           4



OUTPUT AREA INFORMATION

          TOTAL       USED      TOTAL       USED PCNT    AVG     BLOCKS     BLOCKS       TOTAL
AREA     TRACKS     TRACKS     BLOCKS     BLOCKS FULL  SLACK   EXPANDED COMPRESSED     RECORDS

 DEM          2          1         62          2    3    503        N/A        N/A          17
```

❹

The BACKUP FILE INFORMATION consists of the following:

- Job name used to produce the backup
- The backup job's run unit number
- The date and time the backup job was executed
- The CXX name
- The database ID
- The sequence option
- The record ID option of the backed up information
- The area name provided to the backup. If the input to the LOAD function was a backup produced by CA Datacom/DB Version 8.0, the literal "8.0" is printed instead of an area name.

The BACKUP FILE INFORMATION is optional and variable. It is optional in that it is produced only if FORMAT=BACKUP is specified in the LOAD function statement and the input was produced by the CA Datacom/DB Utility (DBUTLTY) BACKUP function. It is variable in that if more than one BACKUP output is provided to the LOAD, each has its control information reported.

Normally one backup is input to the LOAD utility, however this is not always true. Users in all systems sometimes SORT or MERGE backup files to a single file. These functions are normally done when moving tables among different areas for performance or clustering reasons.

If this section is present, it might present one of the following messages:

DB13012E INSUFFICIENT SPACE IN aaa AREA

DB13014 DATA RECORD(S) LOADED WITH DUPLICATE MASTER KEY.

Note: The messages **DB13015W DATA LOADED WITHOUT REORGANIZATION** and **DB13024I DATA LOADED WITH REORGANIZATION** that were printed in prior versions are no longer printed. A backup input with sequence of PHYSICAL and record ID of YES are non-reorganizing. All others reorganize.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                    Copyright © 2009 CA. All rights reserved.                 *               SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX



BACKUP FILE INFORMATION

JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA

DBDVL      56,377   5/25/2003 16.26.35  PRODCXX       1  PHY   NO    DEM


 ❺                              ❻

INPUT FILE INFORMATION          OUTPUT TABLE INFORMATION

AREA  TABLE  ID     RECORDS     AREA  TABLE  ID     RECORDS     NOTES

DEM   PNC     6           5     DEM   PNC     6           5
DEM   PNM     5           3     DEM   PNM     5           3
DEM   POH     3           5     DEM   POH     3           5
DEM   POL     4           4     DEM   POL     4           4


 ❼

OUTPUT AREA INFORMATION

          TOTAL       USED      TOTAL       USED PCNT    AVG     BLOCKS     BLOCKS       TOTAL
AREA      TRACKS      TRACKS    BLOCKS      BLOCKS FULL  SLACK   EXPANDED COMPRESSED     RECORDS

 DEM          2           1         62           2    3    503        N/A        N/A          17
```

❺

The INPUT FILE INFORMATION has a line for each unique table ID of input records. This part prints the area name, table name, table ID, and number of records.

For partitioned tables, provides the partitioned table name and the Full Parent table name.

For a given table loaded, if input records existed, they are reflected on the same line as the output table. If input records were found for a table not being loaded, then they are printed on separate lines with a note of TABLE ID NOT LOADED. These records were read by the LOAD utility but are ignored.

This section for input records only has information printed if the input was from execution of the BACKUP utility function. Extract data or DBFLSUB exit data does not have this information. Records for a single table ID could be input from multiple original backups. All of these records are loaded, however, if the table name or area name from the input stream is different, the name printed on the report for these records is three question marks (???). This does not mean that an error exists, but that the records came from different sources with different area or table names.

If a table is loaded with zero records, and if the input also reflects the table and zero records, a note of TABLE EXCL OR NOT INCL prints, if the table was excluded, or if an include was done but not for this table. If the note is blank, the table had no records at the time of the backup.

❻

The OUTPUT TABLE INFORMATION has a line per table loaded with the area name, the table name, the table ID, and the number of records.

For partitioned tables, provides the partitioned table name and the Full Parent table name.

The following information is provided under NOTES on the load report when tables are partitioned:

SKIPPED

Indicates record(s) from this table are to be skipped (not loaded) because they do not meet any of the range criteria for any tables in this partition.

TABLE NOT LOADED

Indicates the table was not loaded.

❼

The OUTPUT AREA INFORMATION prints the area name, total tracks, used tracks, total blocks, used blocks, percent full, average slack, blocks expanded, blocks compressed, and total records for the area loaded.

The used tracks, used blocks, and percent full reflect high-water marks. If slack space allows for records to be added, or if the backup was physical with record IDs then additional space may exist in this area for record additions. CXX reporting reflects embedded free space. The blocks expanded and blocks compressed require additional CPU time to build and maintain and are requested using the STATS parameter for the load. If STATS are not requested, N/A is printed in these columns.

Chapter 21: LOCK (Lock a Data Area)

The LOCK function allows you to back up a data area, specifically a compressed data area, while the Multi-User Facility is running. By granting this request, DBUTLTY inhibits an update that moves data records until the UNLOCK function is issued. See UNLOCK (Unlock a Data Area) (see page 719) for information.

A lock, when granted, remains in effect until an unlock is done or the Multi-User Facility is terminated. If you request a lock on an already locked area, your request waits until the first lock is released and then is granted, thereby locking the area again.

Note: Any Index Area and data area buffers with pending writes are written to DASD for the specified area when this function is executed.

LOCK

The LOCK function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

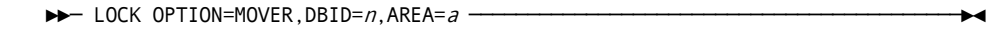
When to Use

When backing up or performing maintenance on a data area, use the LOCK function to prevent movement of records in compressed data areas.

Although this function is designed specifically for compressed data areas, you can use LOCK on any data area.

How to Use

Multi-User mode is required to execute this command. Use the following command format to issue a LOCK function.



Command

LOCK

Invokes the function that inhibits an update until the UNLOCK function is issued.

Required Keywords

OPTION=MOVER

Requests the option that inhibits movement of data records due to an update of a compressed data area.

,DBID=

Identifies the database you want to lock or protect during the backup.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,AREA=

Identifies the area you want to lock or protect during the backup.

Valid Entries:

DATACOM-NAME of the area in the database specified

Default Value:

(No default)

Example JCL

The following shows the command to lock a data area during a backup.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and .
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          LOCK      OPTION=MOVER,DBID=1,AREA=DEM
 /*
 /&
 * $$ EOJ
```

Sample Report: LOCK

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             LOCK      OPTION=MOVER,DBID=1,AREA=DEM

                    FUNCTION=LOCK
       ❷               AREA=DEM
                       DBID=1
                       OPTION=MOVER

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************
                                                                                                    ❹

      ❺             *                                    REQUEST COMPLETE
```

❹

The name of the Directory you are using.

❺

Message indicating that the LOCK completed successfully.

Chapter 22: MASSADD (Add Records to Table)

MASSADD

The MASSADD function communicates with one Multi-User Facility. You can use it with more than one Multi-User Facility when you specify MULTUSE=YES. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When to Use

Use the MASSADD function to add a large volume of records to a table. Some application systems can be designed to accumulate records during daily online processing that are to be added in a single batch run during non-prime hours. You can use MASSADD to perform this maintenance without having to write a specialized application program.

Use of Pipeline

MASSADD takes advantage of the Pipeline high-performance maintenance feature of CA Datacom/DB. The function issues warning messages if logging is not active. See the section on the Pipeline in the *CA Datacom/DB Database and System Administration Guide* for details.

Log Check Points and Backout

In the Multi-User Facility, when logging is on for this table, MASSADD issues a LOGCP every 1000 records and displays a message with a record count each time a checkpoint is taken. The function generates a User Requirements Table with transaction backout specified. MASSADD offers a restart capability to allow you to begin processing the input records at the exact point of any failure.

Unique Keys

The MASSADD function terminates with error message DB13037E identifying the record, a return code 10, and a small snap dump, if you attempt to add a record with a duplicate value for a key which is defined to be unique (see the chapter on using dumps in problem determination in the *CA Datacom/DB Database and System Administration Guide* for information on reading the snap dump). Transaction backout is not invoked when this condition is encountered. You can, however, execute the MASSADD function with the STARTAFT= parameter which allows you to specify a number of records to skip in the input data set.

Integrity Constraints

When executing through the Multi-User Facility, the MASSADD function enforces any integrity constraints defined for a table.

The MASSADD function terminates with an error message, if you attempt to add a record which contains invalid data for any field which is defined in CA Datacom/DB with DBEDITS=Y.

If any field in the table is defined in CA Datacom/DB with FORCEADD=Y attribute-value, the MASSADD function is treated as an application adding records to the table and new values are set for the fields.

Use of the Multi-User Facility

MASSADD can perform the add requests either through the Multi-User Facility or as a Single-User task. You must run MASSADD through the Multi-User Facility to add records to a table defined with integrity constraints.

We recommend that you run MASSADD with the Multi-User Facility to take full advantage of the normal RESTART and RECOVERY options, since the logging facility is not available in Single User mode. If you use Single User mode, checkpoint messages do not occur and transaction boundaries are not provided. Also, if the utility fails, a log file does not exist to be restarted to insure that the data and index are synchronized. If you do not use the Multi-User Facility, back up the area before running the MASSADD function. This way you can reload the backup and try again if the function does not complete.

Tables Using DBVVRPR

The MASSADD function does not support adding records to tables defined with a user compression exit name (DBVVRPR).

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any Multi-User Facility for read.
- The database may *not* have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO*

- The database will be opened for update.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when MULTUSE=YES*

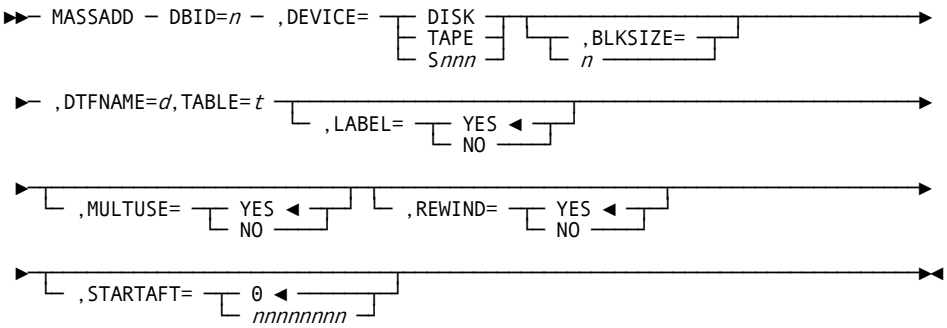
- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The area may *not* have BACKUP, EXTEND, EXTRACT, INIT, LOAD, RECOVERY, REORG, REPLACE, or RETIX with MULTUSE=YES executing.

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility does *not* set ACCESS area.
- The utility does *not* set MASSADD executing this Multi-User Facility this database or area.

How to Use

MASSADD can run in Single User mode if NO is specified for the MULTUSE= keyword. Execute DBUTLTY using the following command format:



Command

MASSADD

Invokes the function to accept input records for a single table and add those records to the table.

Required Keywords

DBID=

Specifies the database identifier for the database containing the table to which the records are being added.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,DEVICE=

Specifies the device type for the input data set.

When DEVICE=TAPE is specified, the system logical unit (SYS) number must be 5 (SYS005).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

,DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the input data set. If DEVICE=TAPE, this name must match the name which you code in your z/VSE JCL TLBL statement; otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

,TABLE=

Specifies the table in the database to which the records are to be added.

Valid Entries:

DATACOM-NAME of the table in the database specified

Default Value:

(No default)

Optional Keywords

,LABEL=

Specifies whether or not tape labels are to be processed. We recommend that labels always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,BLKSIZE=

CA Datacom/DB reads the block size value specified on the input file and ignores a valid BLKSIZE= value assigned in a MASSADD function. We recommend not using the BLKSIZE= keyword. This keyword was previously required and therefore, for convenience, CA Datacom/DB continues to accept it and check for a valid value. Should an invalid value be assigned to the BLKSIZE= keyword, CA Datacom/DB returns an error message.

Valid Entries:

1 to 32767

Default Value:

(No default)

,MULTUSE=

Specifies whether the add commands are to be directed to the Multi-User Facility.

YES

MASSADD directs its input to the Multi-User Facility. Direct commands to the Multi-User Facility when adding records to a table defined with integrity constraints.

NO

MASSADD executes as a Single User job. In Single User mode, logging is not supported, and you do not have a log file to restart to ensure that the data and Index Areas are synchronized. Before using MASSADD, obtain a backup of the area in case you need to reload and try the function again.

Valid Entries:

NO or YES

Default Value:

YES

,REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates the tape is to be rewound before the OPEN and after the CLOSE. NO indicates no rewind before the OPEN or after the CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

,STARTAFT=

Specifies a number of records to skip in the input data set before beginning to add records.

Valid Entries:

Any 1- to 8-character integer

Default Value:

0

Example JCL

The following shows the command to add records to table TEL in database 400. The input data set is TELLER. In this example, the job defaults to Multi-User Facility mode.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYS005,TAPE
 // PAUSE MOUNT TAPE
 // TLBL TELLER,'input'                              Input data set
 // EXEC DBUTLTY,SIZE=750K
         MASSADD   DBID=400,DEVICE=TAPE,DTFNAME=TELLER,TABLE=TEL
 /*
 /&
 * $$ EOJ
```

Sample Report: MASSADD

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2014 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             MASSADD   DBID=400,DEVICE=TAPE,DTFNAME=TELLER,TABLE=TEL

                    FUNCTION=MASSADD
       ❷               DBID=400
                       DEVICE=TAPE
                       DTFNAME=TELLER
                       TABLE=TEL

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2014 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    RECORD ******1000 SUCCESSFULLY ADDED
               ❹     RECORD ******2001 SUCCESSFULLY ADDED
                    RECORD ******3000 SUCCESSFULLY ADDED

                                   ADDED

               ❺         TEL         3,400
```

❹

A line is produced for each 1,000 records successfully added.

❺

Report indicates that 3,400 Records were added to the table.

Chapter 23: OLREORG (Reorganize a Database Online)

When the physical sequence of records in an area is causing a performance problem, a database reorganization is usually scheduled. In the past, this would require the table being rendered unavailable while backups and reloads of the data in Native Key sequence are performed. With OLREORG, however, you can reorganize the database table without having to take it offline.

For information on DBUTLTY STATUS console command support for this function, see DBUTLTY STATUS (see page 37).

Note: DBUTLTY status is only available when the FIRSTKEY= and LASTKEY= keywords are *not* specified or no rows are found in the specified key range.

OLREORG

The OLREORG function of DBUTLTY combines rows in a given key range (reference group) into as few blocks as possible while reducing the effort (physical I/O) needed to retrieve the rows in native sequence, and it does this without making the data unavailable to other users. OLREORG moves rows from blocks with the fewest rows within a specified reference group (REFGROUP) to blocks with the most rows within the REFGROUP, thereby reducing the number of blocks used to contain the REFGROUP. Be aware, however, that OLREORG does not provide a one hundred percent Native Key sequence.

A reference group is a specified number of data blocks that are to be considered as a group in analyzing (reporting) and reorganizing (using OLREORG) data rows on a reference group by reference group basis. You can use statistical information on sequential data access to determine if there is a significant amount of sequential access, and of those accesses how many are to non-contiguous data blocks.

Once you have established that a given table has significant sequential access where there have been a large number of non-contiguous data block accesses, or if you know that a significant amount of sequential processing is to occur, you can execute the DBUTLTY REPORT REFGROUP option to review the current native sequence population. For information on the REFGROUP report, see REFGROUP Report (see page 549).

Once your review of the current native sequence population is completed, you can decide if an online reference group reorganization should be executed to improve the native sequence population.

OLREORG runs through the Multi-User Facility so that concurrent access is available to other users of the specified table.

Note: You can interrupt or stop the execution of OLREORG without any risk to your environment by issuing a COMM REQABORT. The REQABORT is honored after each reference group is processed. In this case, no summary report is produced. The last key value processed is provided instead. This key value can be used for the FIRSTKEY= parameter if you want to start the process again by running a new DBUTLTY OLREORG.

If the OLREORG is not able to be interrupted using REQABORT, an operator cancel can be used.

For verification of the reorganization process, you can include another reference group (REFGROUP) report specification after the reference group reorganization request to see the effect of the OLREORG.

Restrictions

The following restrictions apply:

- The hardware environment in which the OLREORG function is run must have relative branching support.
- The area in which the table (that you specify for OLREORG) resides must be loaded with URI=YES.
- Logging must be turned on in order to use OLREORG.

Security Considerations

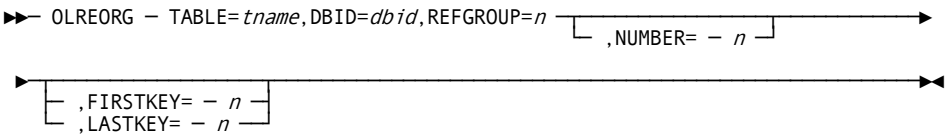
The DBUTLTY external resource name for the OLREORG function is OLREORG. It has no table rights associated with it.

When to Use

Execute the DBUTLY OLREORG function when you determine that a reference group reorganization is appropriate.

How to Use

Multi-User mode is required to execute this command. Execute the OLREORG function using the following command format.



Command

OLREORG

Invokes the OLREORG function.

Required Keywords

,TABLE=*tname*

Specify the name of the table that you want to use. The area in which the table resides must be loaded with URI=YES. If you attempt to generate a REFGROUP report and the area was not loaded with URI, an error occurs and a return code is generated.

Valid Entries:

a valid table name

Default Value:

(No default)

,DBID=

Specify the name of the DBID that you want to use.

Valid Entries:

a valid DBID

Default Value:

(No default)

,REFGROUP=*n*

Specify the number of blocks to treat as a reference group. Each REFGROUP represents the number of blocks which are treated as a group. Each REFGROUP also has a certain number of blocks in which it already has rows stored. The goal of OLREORG is to move rows from blocks with the fewest rows within the REFGROUP to blocks with most rows within the REFGROUP, and hence to reduce the number of blocks used to contain this REFGROUP.

A REFGROUP value represents the number of data blocks it takes to hold a range of key values for the native sequence key. For example if you specify REFGROUP=5, the index is scanned starting with the lowest key, and the data block to which each key value points is noted. This scanning continues until more than 5 unique data blocks are encountered. These 5 unique data blocks and their associated index values represent the first REFGROUP. The 6th data block and its associated key value represent the start of the next REFGROUP.

You must consider what seems to you a good value for REFGROUP=. If you commonly use the DBURTBL macro parameter SEQBUFS= to read this table, half of the value you specify for SEQBUFS= may be a good value for REFGROUP=.

Valid Entries:

2 through 100

Default Value:

(No default)

Optional Keywords

,FIRSTKEY= and/or ,LASTKEY=

You can use FIRSTKEY= and/or LASTKEY= to limit OLREORG to a segment of the native sequence key. We recommend you allow FIRSTKEY=/LASTKEY= to default. The OLREORG function normally defaults to include the full specified key range from low values to high values. FIRSTKEY= overrides the default starting position while LASTKEY= overrides the default ending position, limiting what is processed by OLREORG. If one keyword is specified and the other is not specified, a default value is selected for the missing keyword. Normal CA Datacom key lengths are from 1 through 180 bytes. The key value data entered for FIRSTKEY= and LASTKEY= can be from 0 (zero) through 59 bytes long. You only need to code the number of positions of the key value that are significant to you, but be aware that the value coded is *left justified regardless of key field data type.* The utility pads the low order or remaining positions of the key value with low values on FIRSTKEY= and high values on LASTKEY=. To use the entire key in one step, specify neither FIRSTKEY= nor LASTKEY=. Alternately, you could specify FIRSTKEY=00 and/or LASTKEY=00 to cause processing to start at low values and end at high values. The length of zero indicates that the padded key values (low values and high values) are to be used.

The data is not edited or interpreted. Therefore, if the key value data you need to enter is in binary, you must set up the JCL using a hexadecimal mode display and enter the key value data in hexadecimal. This data can contain blanks, commas, and any other special characters. CA Datacom/DB cannot edit this data for syntax, so the control statement can *look* invalid, especially when it includes blanks or commas or key value data containing English words. For example, FIRSTKEY=12121212121212 means the length is 12, key value is 121212121212. Another example, FIRSTKEY=10FIRSTKEY10 means the length is 10, key value is FIRSTKEY10.

Valid Entries:

A 2-byte length (in a range of 00 through 59) followed by a character string of key value bytes (for the length specified) that represents the key value.

Default Value:

The starting/ending position of the full key range from low to high values.

NUMBER=*n*

This parameter can be specified as 1-99 or omitted. When you omit NUMBER=, all blocks qualify as source blocks and OLREORG moves as many rows as possible, so that as few blocks as possible contain the REFGROUP. Specifying a value for this parameter indicates you want to control which blocks qualify as move-from blocks. It can be used to limit row movement to those cases which provide the most benefit. For example, specifying NUMBER=2 indicates that only blocks with two or fewer rows may have their rows moved to other blocks in the REFGROUP.

Valid Entries:

1 through 99

Default Value:

(No default)

Example JCL

Following is example JCL for an OLREORG.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see page 34).
// ASSGN  SYSnnn,DISK,VOL=volser,SHR
// DLBL   IXX781,'ixx.db781',,DA
// EXTENT SYSnnn,volser
// DLBL   781A01,'A01.db781',,DA
// EXTENT SYSnnn,volser
// EXEC   DBUTLTY,SIZE=768K
   OLREORG TABLE=A01,DBID=781,REFGROUP=50
/*
/&
* $$ EOJ
```

Sample Report

For sample REFGROUP reports, see Sample Report: REFGROUP Summary (see page 553). An example of the summary report generated by executing the OLREORG function against a table containing 3,000 rows follows:

```
Rows processed -                 3,000   ❶
Rows moved -                       384   ❷
Blocks freed in reference group -  129   ❸
Exclusive control conflicts -        0   ❹
```

❶ Rows processed is the total rows in the table, or if using FIRSTKEY or LASTKEY, the total rows within the specified key range.

❷ Rows moved is the total rows moved from less heavily populated blocks within a reference group to more heavily populated blocks.

❸

The number printed for the blocks freed in reference group is not an *absolute* number, but must be understood as being relative to the reference group concept. That is, at the start of processing each reference group uses the same number of blocks specified by the REFGROUP parameter. After the OLREORG is run, the same set of rows for each REFGROUP may be in fewer blocks. A block freed in a reference group represents the difference in the number of blocks required to hold each reference group. Because each actual physical block may contain rows from multiple REFGROUPs, the number printed in the report does *not* therefore necessarily represent completely *empty* blocks, that is, a block *freed* is not necessarily the same as an *empty* block.

❹

Exclusive control conflicts refers to the following. Applications are allowed to maintain rows in the specified table while OLREORG is running. The number printed in the report represents rows that OLREORG chose for movement but skipped because some other task currently had either primary or secondary exclusive control on the row. These rows are skipped for performance reasons.

Chapter 24: RECOVERY (Rebuild a Database)

The RECOVERY function of DBUTLTY rebuilds a damaged database from data in the Recovery File (RXX).

We recommend that you use inactive recovery (see the information about the LOGRCV Multi-User startup option in the *CA Datacom/DB Database and System Administration Guide*).

Types of Recovery

There are two RECOVERY options available:

Forward

Reapplies maintenance transactions as they are recorded in the Recovery File (RXX). Use forward recovery to bring a database up to date when a restore of a previous backup has been necessary.

Backward

Reverses maintenance transactions recorded in the Recovery File (RXX). Use backward recovery to reverse or back out maintenance that was applied.

Note: Because SQL DDL statements are not recorded in the Log Area, they are not recoverable using the RECOVERY function of the DBUTLTY. In the case of DDL statments, it is therefore your responsibility to ensure the existence of the Directory (CXX) defintions necessary for recovery.

When to Use

Use forward recovery when your database has been partially or fully destroyed or is unusable. Use backward recovery when you want to reverse transactions from the current status of the database.

If MULTUSE=YES has not been specified, no other task can be processing the database when you are running foward RECOVERY.

RECID= Keyword

The optional keyword RECID=, when set to NO, allows forward recovery to be based on the Master Key instead of the row identifier. Using RECID=NO is not for "normal" recovery applications. For more information, see the table of Optional Keywords.

Impact of Data Area Reorganization

The forward and backward recovery operations require that record location information in the database match the record location information that existed at the time the maintenance was performed. A reorganization of the database causes new record location information to be assigned. Therefore, any reorganization causes Recovery File(s) (RXX) created prior to the reorganization to be unusable.

RXXs without End-of-File or Trailer Records

If the operating system or the Multi-User Facility terminates abnormally while CA Datacom/DB is writing to a Recovery File (RXX), the tape has no End-of-File or trailer records.

When you restart the Multi-User Facility, DBUTLTY detects the condition and reports the last completed cycle number. The restart process then changes the cycle and the log blocks being spilled from a *being spilled* status to a *needing to be spilled* status. A later spill can then copy these blocks in their entirety.

Two problem situations exist at this point:

- Because no trailer records exist on the active recovery tape, recovery processing with this tape is a problem. Backward recovery cannot function at all because the trailer records cannot be found. Forward recovery can function, but the recovery receives an I/O error.
- The second problem is that when executing forward recovery, the next Recovery File after the one with no End-of-File, detects errors because of the duplicated log records which require review.

To eliminate these problems, execute the RXXFIX function RXXFIX (Recover End-Of-File) (see page 691).

Multiple File Recovery Files (RXX)

When the Recovery File (RXX) input to be used for forward recovery includes multiple files, in z/VSE the INNUM= keyword must be used to specify the number of input RXX files. When end-of-file is detected, DBUTLTY checks the INNUM= keyword value to see if additional files are required. If they are, the current RXX file is closed and unloaded and an OPEN issued for the next file. This OPEN causes a message, 0P08t INTERV REQ, indicating the next file needs to be mounted. When the new file is opened, you may receive a message: 4132D ERROR IN FILE ID. If the correct tape is mounted, reply IGNORE.

Note: When CA-DYNAM is active, the processing option U (for unload) must be specified to allow this facility to work properly. Errors result if U is not specified, because without U specified, the tape is rewound and the next open causes the same file to be read a second time.

Backups for Use with Forward Recovery

To use the CA Datacom/DB forward recovery to recover a data area, the records in the database must be restored to the same physical locations at which they resided at the time the recovery records were logged. You can obtain a backup that contains the records that can be restored correctly in one of two ways:

- Use the BACKUP Data Area option with SEQ=PHYSICAL and RECID=YES to create a backup in physical sequence with record location information.
- Create a backup in Native Sequence and immediately restore it. This reorganization invalidates any Recovery Files (RXX) that were made prior to the backup. However, any Recovery Files created beyond this point can be used to restore the database after reloading this backup.

A physical sequence backup can also be obtained using any other physical backup utility. This type of backup cannot be used directly as input to the DBUTLTY LOAD function. After restoring data from this type of backup, the Index can be restored using the DBUTLTY Reconstruct Index (RETIX) function.

Note: Forward recovery expects to be executed after an area restore. It is not intended to be executed after a backward recovery. This requirement is especially true when running without the URI option.

Forward recovery into areas that have been dynamically or manually extended require that the data set size reflect all tracks that previously existed. Forward recovery into an area that does not reflect all extended tracks may receive a return code 94(086). This is true when running the RECOVERY function with the MULTUSE= keyword set to YES or NO. A forward recovery execution does not successfully dynamically extend a data set. The 94(086) retun code should occur. Log records used in this recovery process could contain recovery information from the missing extended tracks causing the error.

Continuous Processing

To continue processing during a backup, you must use the physical backup method with UPDATE=NO specified. If maintenance processing occurs during the backup, error messages are to be expected when a subsequent forward recovery is used to restore the database. You should only receive errors for maintenance that occurs during the time span that the backup was run. Carefully examine these errors to be sure that all maintenance is properly restored. For more information, see BACKUP (Create Backups) (see page 89) of this manual and continuous processing in the *CA Datacom/DB Database and System Administration Guide.*

Backward Recovery

Backward recovery requires no database preparation. The backward recovery process alters the database by reversing maintenance transactions, returning it to its unchanged state.

RECOVERY OPTION=BACKWARD performs by reading the selected data forward using the name RXX and sorting it into reverse sequence.

You can use the STIME= and ETIME= keywords to reverse part of a job, but this is not a recommended technique. Examine any backward recovery, other than an entire job, carefully.

Multi-User Facility Connection

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

Impact of Integrity Constraints (Forward Recovery)

The impact of integrity constraints on the forward recovery process depends on whether all dependencies of all tables in the database are contained within that database. During the recovery process, if all dependencies of all tables are contained within the database in question, and you have **not** requested individual table recovery or selected a subselect of the recovery (such as, by job name), the system assumes that integrity is not violated.

In all other cases, tables defined with integrity constraints which are processed by the recovery function are marked CHECK-PENDING, indicating that referential integrity must be checked (see CONFIRM (Verify Constraint Integrity) (see page 199). Applications attempting to access such tables may receive a return code, as indicated in the following chart:

For forward recovery:	Then:
If a table references another table (is a child)	DELETES are allowed, but ADDS and UPDATES generate a CHECK-PENDING return code.
If a table is referenced by another table (is a parent)	ADDS are allowed, but DELETES and UPDATES generate a CHECK-PENDING return code.

Note: You must CONFIRM tables in the following order, from referenced-to-referencing, or parent-to-child, that is to say, the referenced table cannot be in check pending status. For example, if there is a foreign key from a REMARKS table to a LINE_ITEM table on PO_Nbr and Line_Item_Nbr, and the LINE_ITEM table has a foreign key that references the PO table on PO_Nbr, the PO table must be loaded first, then the LINE_ITEM table can be confirmed, then the REMARKS table can be confirmed. Therefore, the confirm order would be: from PO table to LINE_ITEM table to REMARKS table.

Impact of Integrity Constraints (Backward Recovery)

Backward recovery exists in the Multi-User Facility with all normal constraint checking in force.

Index Option

For all backward recoveries and most forward recoveries, it is most efficient to perform Index maintenance with the data maintenance. That is, for each record added, the required Index entries are also added. At the completion of the recovery, all work is done and the data and Index are synchronized.

In special situations where most of an area is added or deleted (or updated with key changes), this conventional way of handling Index maintenance may not be the fastest process. The conventional process also may not be the best if key fields are frequently changed during the day and for tables with many keys.

With the INDEX= keyword, you specify whether the backup is loaded in the conventional manner or with no Index maintenance. You cannot run the function with an initialized-only Index or a wrong Index. The URI Index and space Index are required to be accurate. After loading the data areas, the forward recovery would also be done without Index work (INDEX=NO). At the completion of the recovery, you run the RETIX function on each database recovered to build its Index for the data records.

How to Use Backward Recovery

Multi-User mode is required to execute this command. To use the DBUTLTY RECOVERY function for backward recovery, submit at least the RECOVERY command. The RECBASE command is optional. If you specify a RECBASE command, you limit the recovery to the tables in a single database or to a specific table, otherwise the RECOVERY function opens tables as necessary. If you do not specify an optional RANGEDT keyword, *all* the jobs accessing the DBID specified in the RECBASE command(s) are recovered. Do not specify a RANGEDT keyword for continuous operations.

Backward Recovery Successful Execution Requirements/Controls

Environmental **Requirements**

Note: Rules are the same as a user application opening a table to update records.

- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The area may *not* have BACKUP, EXTEND, EXTRACT, INIT, LOAD, RECOVERY, REORG, REPLACE, or RETIX with MULTUSE=YES executing.

Environmental **Controls**

- The database will be opened by Multi-User Facility for update.
- The utility does *not* set ACCESS area.
- The utility does *not* set RECOVERY executing this Multi-User Facility this database or area.

How to Use Forward Recovery

Forward recovery can be run in a Multi-User mode or in a Single User mode while the MUF is up, providing that the database is closed and not being used. To use the DBUTLTY RECOVERY function for forward recovery, you must submit at least one RECBASE command and only one RECOVERY command. You can use an optional RANGEDT keyword to define a *date/time-range.* The first date/time is the lowest time to process and the second date/time is the highest time to process.

The JOBNAME= keyword allows you to specify up to 100 RECJOB commands.

If you do not specify a RANGEDT keyword, *all* the jobs accessing the DBID specified in the RECBASE command(s) are recovered. For continuous operations, do not specify a RANGEDT keyword.

Impact of Table Partitioning

For information about the impact of table partitioning on the forward recovery function, see Forward Recovery Utility (see page 42).

Forward Recovery Successful Execution Requirements/Controls

Environmental **Requirements** *when MULTUSE=NO*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any Multi-User Facility for read.
- The database may *not* have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO*

- The database will be opened for update.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when MULTUSE=YES*

- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.

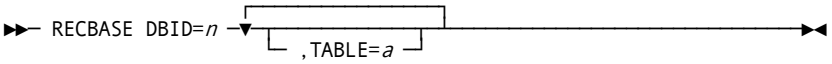
- The area may have no table in an open User Requirements Table.
- The area may *not* have any other MULTUSE=YES utility executing.
- The area must be closed.

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets RECOVERY executing this Multi-User Facility this database or area.

RECOVERY Command 1 (RECBASE)

The RECBASE command is optional for backward recovery and required for forward recovery. You can use it to specify an entire database or the specific table(s) for recovery. If you do not code a RECBASE command, the backward RECOVERY function dynamically opens tables as needed.



Command

RECBASE

Specifies the database and tables for recovery.

Required Keyword

DBID=

Identifies the database to be recovered.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Optional Keywords

,TABLE=

Specifies a table for recovery. Name one table with each TABLE= keyword.

Valid Entries:

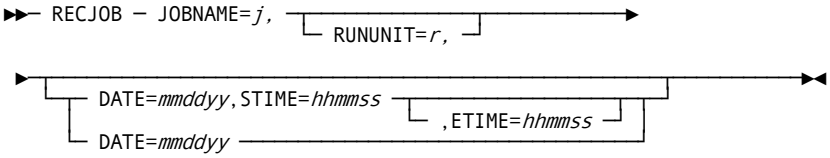
DATACOM-NAME of the table in the database specified

Default Value:

If this keyword is omitted, CA Datacom/DB recovers the entire database.

RECOVERY Command 2 (RECJOB)

The second command, RECJOB, provides details concerning each job you want recovered or a date/time for all jobs. The JOBNAME= keyword allows up to 100 RECJOB commands to be specified with a single RECOVERY function.



Command

RECJOB

This is an optional command to specify details concerning each job. The JOBNAME= keyword allows you to specify up to 100 RECJOB commands.

Required Keyword

JOBNAME=

With a RECJOB command, the JOBNAME= keyword identifies the job for which recovery is requested. The JOBNAME= keyword allows you to specify up to 100 RECJOB commands.

Valid Entries:

Any valid job name for recovery

Default Value:

(No default)

Optional Keywords

DATE=

Specifies the date for the job to be recovered. Obtain this information from the timestamp on the console. When you include STIME= you must also include this keyword. However, you can use this keyword by itself.

Valid Entries:

A date in the form: mmddyy (month-day-year)

Default Value:

(No default)

,ETIME=

Specifies the ending time for the recovery. Obtain this information from the timestamp on the console. When you include this keyword, you must also include the STIME= and DATE= keywords. If you use the job date and/or time keywords for a job spanning two days, include two date/time keywords. See the second example at the end of this chapter. Do not use this keyword if you want to recover the entire job.

Valid Entries:

A time in the form: hhmmss (hours-minutes-seconds)

Default Value:

If STIME= is specified, the default is 235959. Otherwise, no default.

RUNUNIT=

Identifies the run unit, by number, assigned to the job to be recovered. CA Datacom/DB displays the number of the run unit, which is assigned to a job, on the console. The run unit is used to identify which iteration to recover when there are multiple runs of a single job name on the Recovery File.

Valid Entries:

Run unit number

Default Value:

(No default)

,STIME=

Specifies the start time for the recovery. Obtain this information from the timestamp on the console. When you include this keyword, you must also include the DATE= keywords. Do not use this keyword if you want to recover the entire job.

Valid Entries:

A time in the form: hhmmss (hours-minutes-seconds)

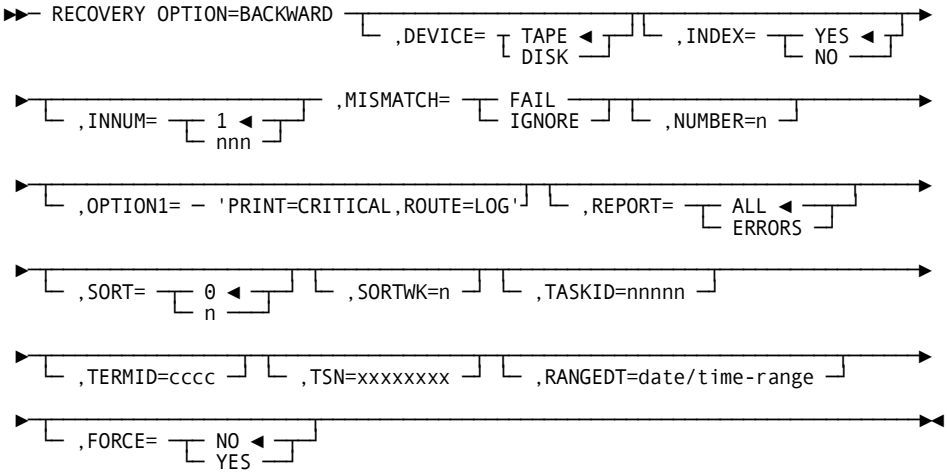
Default Value:

(No default)

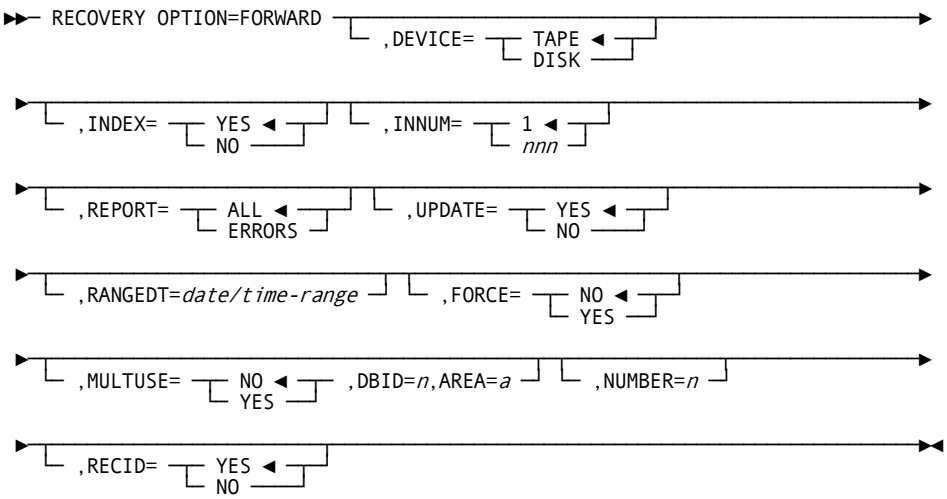
RECOVERY Command 3 (RECOVERY)

The RECOVERY command requests that CA Datacom/DB perform a database recovery. All applicable RECBASE and RECJOB commands must immediately precede this command.

Backward Recovery



Forward Recovery



Command

RECOVERY

Invokes the function to rebuild a damaged database.

Required Keyword

,MISMATCH=

(For backward recovery only.) Specify the action the RECOVERY function is to take in the event that the data has been modified since the commit. FAIL means to stop and invoke rollback, while IGNORE means to skip the offending update and continue with the recovery. In both cases, the failing update is not applied to the database.

A MISMATCH for the specific maintenance commands is defined as follows:

Logged DELET Record

The record is found. For URI areas (recommended), the right URI record is found in the required location (block), or a moved record index (URI index) exists indicating another location where the record is found. For non-URI areas (not recommended), the right record is simply any record found in the required block that currently contains the logical record number (1-255). Sufficient DELET/ADDIT combinations can cause the numbers to be reused when space is reclaimed.

Logged UPDAT Record

First case, the record is not found. Second case, the record found elements mismatch with the elements from the logged UPDAT record. Note that only the elements in the UPDAT command are compared, and if they are equal the update restores only those elements to the before update state.

Logged ADDIT Record

First case, the record is not found. Second case, the record found elements mismatch the elements from the logged ADDIT record. Note that if the elements match, the row is deleted. This takes into account no other fields in the record.

Valid Entries:

FAIL or IGNORE

Default Value:

(No default)

OPTION=

Indicates the type of recovery desired.

Valid Entries:

BACKWARD or FORWARD

Default Value:

(No default)

Optional Keywords

,AREA=*a*

Specifies the area, where *a* is the area name, for area level DBUTLTY control.

To use area level DBUTLTY control, three keywords must be provided, MULTUSE=YES, DBID=*n,* and AREA=*a.* The three are a set, that is, if one of the three is specified, the other two must also be specified. When all are provided, the RECBASE and RECJOB functions are not allowed. Using the three area level control keywords, recovery works at the same level as the LOAD, RETIX, INIT, EXTEND, and REORG utility functions.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

DATACOM-NAME of the area in the database specified in the DBID= parameter

Default Value:

(No default)

,DBID=*n*

Identifies the database containing the area, for area level DBUTLTY control.

To use area level DBUTLTY control, three keywords must be provided, MULTUSE=YES, DBID=*n,* and AREA=*a.* The three are a set, that is, if one of the three is specified, the other two must also be specified. When all are provided, the RECBASE and RECJOB functions are not allowed. Using the three area level control keywords, recovery works at the same level as the LOAD, RETIX, INIT, EXTEND, and REORG utility functions.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

DEVICE=

Specifies the device type of the input.

Valid Entries:

DISK or TAPE (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

TAPE

,FORCE=

Specifies whether the recovery utility allows input without the first record being a CYCLE START. The RECOVERY utility expects the input to be a Recovery File (RXX) or a CA Datacom Fast Restore file. It verifies this is correct based upon a control block at the start of the file. Valid entries for FORCE= are NO or YES. The default is NO.

Specifying FORCE=YES removes the edit and forces the utilty to process the input to its best ability. We do not recommend specifying FORCE=YES. The FORCE=YES option exists to allow clients who *know* that a recovery *only needs* data on a volume or volumes of a multivolume RXX file, not including the first. I/O can thus be saved but at the risk that none or only some of the necessary maintenance is completed.

When FORCE=NO is specified, the first Recovery File (RXX) presented to recovery must be the first volume of a possibly multivolume data set. This is what recovery expects and what should normally be done.

If time is important and the risk of doing the wrong recovery is acceptable, you can specify FORCE=YES to allow a non-first volume to start recovery, which can save time if that volume contains the required information.

A console message written when the Recovery File was built provides only the date/times of the first and last record and does not indicate where within that span most of the data lies. If FORCE=YES is specified, an additional message **DB13116I RECOVERY FIRST RECORD WITH FORCE, TIME** is printed on the console, even if a CYCLE START is present.

Two console messages track the presence of the first volume of a Recovery File and the end of a Recovery File. The message at the start of a Recovery File is **DB13112I RECOVERY START rrrrr-ccccc.** The message at the end of a Recovery File is **DB13113I RECOVERY END rrrrr-ccccc, TIMES ccyy/mm/dd hhmmss-ccyy/mm/dd hhmmss.** In both messages rrrrr is the Recovery File number and ccccc is the cycle number.

Valid Entries:

NO or YES

Default Value:

NO

,INDEX=

Specifies whether the index is to be processed during record maintenance. If NO is specified, the RETIX function must be performed for every database processed after the recovery operation.

Valid Entries:

NO or YES

Default Value:

YES

INNUM=

Specifies the number of input recovery files for forward recovery. Specify as 1 for backward recovery or let default to 1.

Note: See Multiple File Recovery Files (RXX) (see page 445).

Valid Entries:

A 1- to 3-character integer indicating the number of files to read for forward recovery

1 for backward recovery

Default Value:

1

MULTUSE=

(For area level DBUTLTY control only.) For area level DBUTLTY control, three keywords must be provided, MULTUSE=YES, DBID=*n,* and AREA=*a.* The three are a set, that is, if one of the three is specified, the other two must also be specified. When all are provided, the RECBASE and RECJOB functions are not allowed. Using the three area level control keywords, recovery works at the same level as the LOAD, RETIX, INIT, EXTEND, and REORG utility functions.

MULTUSE=YES allows a single area in a database to be recovered with no disabling of the other areas in the database. With MULTUSE=YES, the area involved must be able to be opened for update in the Multi-User Facility. Once begun, the function prevents other opens for the area. Tables in other areas may be open for other utilities and user applications based upon normal rules. This recovery does not add log records. If it fails, it is necessary to restore the data area again and restart the recovery, the same as would be required when running any of the other forms of forward recovery. In forward recovery, if you specify MULTUSE=NO, or allow the function to default to it, the database involved must be able to be opened for update by the utility address space. This means it is not open for update by any other address space, including a Multi-User Facility. Also, the database should not be open for read, even though CA Datacom/DB does allow it. For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

NO or YES

Default Value:

NO

,NUMBER=

Specifies the maximum number allowed per execution of dumps generated by return code 04(47), received when the record ID in the Request Area contains invalid data or is not in the specified table. See the *CA Datacom/DB Message Reference Guide* for more information on return code 04(47).

Note: Because running forward recovery with UPDATE=NO indicates a *hot backup,* all 04(47) dumps are suppressed. They do not normally occur when all is done correctly, but in testing and not running exactly what is needed, many can occur. Usually, if many occur they fill the PXX and prevent other useful information from being available. Most often a pattern exists, and two dumps would be plenty to find any real error. The number could be set higher, of course, based upon testing needs or as directed by Technical Support.

Valid Entries:

a number in the range of 0 to 9999 specifying the maximum number allowed per execution of dumps generated by return code 04(47).

Default Value:

2

,OPTION1=

(*z/VSE only*) Specifies the options for the associated sort/merge program application.

The values for this keyword must be 44 characters or less, enclosed in quotes. Depending on the features of your installation's sort, the values that apply to this keyword can vary. CA Datacom/DB does not edit anything that you place inside of the quotes.

Valid Entries:

Any values your installation's sort allows. These values must be less than 44 characters and be inside quotes.

Default Value:

CRITICAL for PRINT
LOG for ROUTE

,RANGEDT=

Specify two date/times to provide a range (lowest to highest times to process). The date/times are specified as a 28 character number with the lower date/time immediately followed by the higher date/time. Each date/time is 14 characters in the format 'ccyymmddhhmmss'. For example: '20010110000000020010228235959'. The input is not read beyond one record higher than this ending value.

Note: A century and year combination of less than 1998 is not allowed.

The recovery report detail lines print the date in the format mm/dd/ccyy (see Recovery File (RXX) Analysis Report).

cc

century

Valid Entries: 19 through 39

Default Value: (No default)

yy

year

Valid Entries: 00 through 99

Default Value: (No default)

mm

month

Valid Entries: 01 through 12

Default Value: (No default)

dd

day

Valid Entries: 01 through 31 (regardless of month)

Default Value: (No default)

hh

hour

Valid Entries: 00 through 23

Default Value: (No default)

mm

minute

Valid Entries: 00 through 59

Default Value: (No default)

ss

second

Valid Entries: 00 through 59

Default Value: (No default)

,RECID=

RECID=NO allows forward recovery to be based on the Master Key instead of on the row identifier.

Note: Using RECID=NO is for *special purposes* as described in the following. RECID=NO is *not* for "normal" forward recovery situations.

Forward recovery requires a database to be in the *same place and identity* to execute forward reprocessing successfully. This is not always possible, for example when recovery is associated with the reorganization of a database, which can require closing the database, resulting in an outage of undesireable duration for 24X7 sites.

RECID=NO, basing forward recovery on the Master Key instead of on the record identifier, can be useful as follows:

- It allows recovered backups to be physically different than the original image.
- It can be used when recovery needs to span reorganizations.
- It can be used to keep duplicate query systems in sync.
- It can be used with DASD snapshots for offline reorganization.

To be used successfully, a stable copy of the data in an area must exist in two places. No maintenance must occur at the second location except forward recovery. All maintenance from the primary location must be applied by this RECID=NO forward recovery in ascending order. No mismatch is possible for ADDIT. The command is simply processed. UPDATE and DELET must match byte for byte.

Note: If you specify RECID=NO, *hot backups* do not work. Also, areas must be URI when using RECID=NO. If you specify RECID=NO and areas are not defined as URI, an error occurs and the following message is generated: **DB13001E UNEXPECTED RETURN CODE rr (iii),** where the return code (rr) is 94 and the internal return code (iii) is (123).

Valid Entries:

YES or NO

Default Value:

YES

,REPORT=

Specify ERRORS to report on *only* those records which have errors. Specify ALL to report on *all* records which were recovered.

Valid Entries:

ERRORS or ALL

Default Value:

ALL

,SORT=

Specify this option to speed backward recovery by providing sort a record estimate to sort. That is to say, the value of the SORT= keyword specifies the estimated number of entries to be sorted. DBUTLTY passes the value specified for this parameter to the sort routine. Therefore, the value specified for SORT= can affect the amount of runtime used by the sort, but when you specify any value, including zero, for SORT=, the sort is performed.

Note: Do *not* specify over 99 million unless you verify that the sort package you are using allows a larger number.

Valid Entries:

0 to 99999999999

Default Value:

0

,SORTWK=

Specifies the number of sort work areas.

Note: Do not specify a 2-digit number unless the sort package you are using allows a 2-digit number. Sort packages that do not support a 2-digit number can fail in an unpredictable manner, if you specify a 2-digit number.

Valid Entries:

0 to 99

Note: Do not use two-digit numbers unless the SORT package to be used supports the larger number.

Default Value:

3

,TASKID=

(For backward recovery and valid for CICS transactions only.) Specify that the updates associated with a specific task ID are to be reversed.

Valid Entries:

5-digit number assigned by CICS

Default Value:

(No default)

,TERMID=

(For backward recovery only.) Specify that the updates associated with a specific CICS terminal ID are to be reversed.

Valid Entries:

4-character CICS terminal ID

Default Value:

(No default)

,TSN=

For backward recovery only, specify the transaction sequence number (TSN) associated with updates needing to be backed out. If the Multi-User startup option RXXROLLBACK is YES, CA Datacom/DB issues a console request (DB00109W) to alert the operator of a failed force spilled transaction and the TSN (in hexadecimal).

Valid Entries:

A hexadecimal value from 00000000 to FFFFFFFF

Default Value:

00000000

,UPDATE=

For forward recovery only, if the LOAD was done from a safe backup, that is, from a backup database that was not being updated during the backup, you can either specify UPDATE=YES or allow UPDATE= to default to YES. If maintenance was or could have been occurring to the backup database during the LOAD, specify UPDATE=NO.

Errors are expected after restoring a backup which was taken while the database was open for update and maintenance was occurring, but when the complete forward recovery process is done, the data is correct. When you do not specify UPDATE=NO, each error (mismatch) causes a dump of information to be generated. This causes the amount of output from the recovery to be large. However, dumps are only necessary to help correct problems if the encountered errors are unexpected. Therefore, specifying UPDATE=NO means that you expect errors, that you assume they are going to be correct when forward recovery is complete, and you do not want the dumping of information to occur.

Note: Only specify UPDATE=NO for the condition described above. Unexpected errors should be allowed to generate dumps. Without the dumps, there is no way to find and correct problems.

Valid Entries:

YES, NO

Default Value:

YES

Recovery Examples

Example 1

The following shows the command to reverse the transactions for the PAYCHECK job on database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYS001,TAPE
 // PAUSE MOUNT TAPE
 // TLBL RXX,'rxx'                                  Recovery data set (backward)
 // EXEC DBUTLTY,SIZE=750K
          RECBASE   DBID=1
          RECJOB    JOBNAME=PAYCHECK
          RECOVERY  OPTION=BACKWARD,SORT=1000,MISMATCH=FAIL
 /*
 /&
 * $$ EOJ
```

Example 2

The following shows the command to forward recover all transactions for database 1 on a single input Recovery File.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // ASSGN SYS001,TAPE
 // PAUSE MOUNT TAPE
```

```
// TLBL RXX,'rxx'                                  Recovery data set
// DLBL IXX001,'db001.index',9999,DA               Index Area for DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// DLBL PAY001,'db001.pay',9999,DA                 Data area PAY in DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// EXEC DBUTLTY,SIZE=750K
         RECBASE   DBID=1
         RECOVERY  OPTION=FORWARD,INNUM=1
/*
/&
* $$ EOJ
```

The report generated by this example follows.

Example JCL

The following JCL shows the commands for backward and forward recovery.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value that you must supply. Code all statements to your site and installation standards.

Recovery Backward

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
// DLBL CXX,'cxx',,DA                          Directory data set
// EXTENT SYSnnn,volser
// ASSGN SYS001,DISK,VOL=volser,SHR
// DLBL RXX,'rxx'                              Recovery data set (backward)
// EXTENT SYS001,volser
// ASSGN   SYS002,DISK,VOL=volser,SHR
 // ASSGN   SYS003,DISK,VOL=volser,SHR
 // ASSGN   SYS004,DISK,VOL=volser,SHR
 // ASSGN   SYS005,DISK,VOL=volser,SHR
 // DLBL    SORTWK1,'SORT.WORK.1',0
 // EXTENT  SYS002,volser,1,0,begtrk,ntrks
 // DLBL    SORTWK2,'SORT.WORK.2',0
 // EXTENT  SYS003,volser,1,0,begtrk,ntrks
 // DLBL    SORTWK3,'SORT.WORK.3',0
 // EXTENT  SYS004,volser,1,0,begtrk,ntrks
 // DLBL    SORTWK4,'SORT.WORK.4',0
 // EXTENT  SYS005,volser,1,0,begtrk,ntrks
// DLBL IXX001,'db001.index',9999,DA            Index Area for DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// DLBL PAY001,'db001.pay',9999,DA              Data area PAY in DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// DLBL DEM001,'db001.dem',9999,DA              Data area DEM in DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// DLBL PMF001,'db001.pmf',9999,DA              Data area PMF in DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// EXEC DBUTLTY,SIZE=750K
          RECBASE   DBID=1
          RECJOB    JOBNAME=PAYCHECK
          RECOVERY  OPTION=BACKWARD,SORT=1000,MISMATCH=FAIL
/*
/&
* $$ EOJ
```

Recovery Forward

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
// DLBL CXX,'cxx',,DA                          Directory data set
// EXTENT SYSnnn,volser
// ASSGN SYS001,DISK,VOL=volser,SHR
// DLBL RXX,'rxx'                              Recovery data set (backward)
// EXTENT SYS001,volser
// ASSGN   SYS002,DISK,VOL=volser,SHR
 // ASSGN   SYS003,DISK,VOL=volser,SHR
 // ASSGN   SYS004,DISK,VOL=volser,SHR
 // ASSGN   SYS005,DISK,VOL=volser,SHR
 // DLBL    SORTWK1,'SORT.WORK.1',0
 // EXTENT  SYS002,volser,1,0,begtrk,ntrks
 // DLBL    SORTWK2,'SORT.WORK.2',0
 // EXTENT  SYS003,volser,1,0,begtrk,ntrks
 // DLBL    SORTWK3,'SORT.WORK.3',0
 // EXTENT  SYS004,volser,1,0,begtrk,ntrks
 // DLBL    SORTWK4,'SORT.WORK.4',0
 // EXTENT  SYS005,volser,1,0,begtrk,ntrks
// DLBL IXX001,'db001.index',9999,DA            Index Area for DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// DLBL PAY001,'db001.pay',9999,DA              Data area PAY in DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// DLBL DEM001,'db001.dem',9999,DA              Data area DEM in DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// DLBL PMF001,'db001.pmf',9999,DA              Data area PMF in DBID 1
// EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
// EXEC DBUTLTY,SIZE=750K
          RECBASE   DBID=1
          RECOVERY  OPTION=FORWARD,DEVICE=DISK
/*
/&
* $$ EOJ
```

Sample Report: RECOVERY

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             RECBASE DBID=1

                    FUNCTION=RECBASE
       ❷                DBID=00001


       ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶              RECOVERY OPTION=FORWARD

                    FUNCTION=RECOVERY
       ❷                OPTION=FORWARD


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

                                          DATACOM/DB RECOVERY


    ❹       ❺      ❻         ❼       ❽      ❾      ❿

JOB NAME R-UNIT   DATE       TIME    REQUEST  BASE  BLOCK      URI  AREA RM START OF RECORD       ERROR MESSAGES

DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000B EMP    00011IH000100000000000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000C EMP    00012AS000500000104000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000D EMP    00013AH000110000104000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000E EMP    00014AS000500000104000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000F EMP    00015AS000500000104000
```

❹

The name of the job being recovered.

❺

The run unit of the job.

❻

The date the record was logged.

❼

The time the record was logged.

❽

The contents of the request area.

❾

The database ID.

❿

The block number in hex.

(page 3 continued)

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                                        DATACOM/DB RECOVERY


                                                                ❶     ❷  ❸❹      ❺                  ❻

JOB NAME R-UNIT   DATE       TIME   REQUEST  BASE  BLOCK      URI  AREA RM START OF RECORD       ERROR MESSAGES

DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000B EMP    00011IH000100000000000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000C EMP    00012AS000500000104000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000D EMP    00013AH000110000104000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000E EMP    00014AS000500000104000
DBL2EPAY 23,067 10/03/2003 14.24.48 UPDATPAY    1 00000002 0000000F EMP    00015AS000500000104000
```

❶

The Unique Row Identifier (URI) number in hex. For non-URI areas, the logical record number within the block in hex.

❷

Contains the area that contains the table of the maintenance command. If the table is a Full Parent partitioned table or an Any partitioned table, the area is the specific area in which the record resides.

❸

The R field contains an R if the log record has been rolled backward by transaction backout. It does *not* contain an R if the request was part of transaction backout but was actually done by a backward recovery in support of RXXROLLBACK. The R field is informational and requires no action.

❹

The M field contains an M if the log record represents a record that has been moved from one block to another. This normally only occurs for a compressed table where the record after an UPDAT does not fit back in the same block and is therefore moved to another block. The M field is informational and requires no action.

❺

The first 20 bytes of the user work area.

❻

The error messages produced during recovery.

The page containing the following fields is produced only when you specify a RECJOB command. The fields contain values only when the job specified in the command is found in the Recovery File.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     4
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                DATACOM/DB RECOVERY JOBS NOT FOUND ON RECOVERY FILE


  ❶          ❷            ❸            ❹        ❺

JOB NAME     DATE       START TIME     STOP TIME  RUN UNIT
```

❶

The job name of the job for which recovery was requested.

❷

The date of the job to be recovered.

❸

The start time for the recovery.

❹

The ending time for the recovery.

❺

The run unit assigned to the job to be recovered.

The page containing the following fields is produced for all jobs.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     5
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

     ERROR SUMMARY                    ❶

TOTAL ERRORS ENCOUNTERED-----0

FIRST 01 PAGE(S) CONTAINING ERROR(S)-----4

LAST ERROR ON PAGE      -----4
```

❶

A summary of the errors encountered.

Chapter 25: REMOVE (Remove Key ID)

The REMOVE function processes the Index Area for the specified database and deletes the entries for the specified key ID.

REMOVE

You cannot perform the REMOVE function while any other task is processing the database.

When to Use

When you change a key definition's length, the old key information must be removed from the Index before rebuilding the Index with the LOAD or Retrieve Index (RETIX) functions. A key's information can be removed from the Index using the REMOVE function, or by initializing the Index. In most cases, REMOVE is preferred because DBUTLTY does not have to rebuild the remaining key definitions when the LOAD or RETIX is executed.

When a large database is involved, executing REMOVE for the key ID(s) whose length(s) changed followed by LOAD or RETIX is generally faster. This is because the Index is rebuilt in a merge operation; so, DBUTLTY does not have to rewrite entries for key IDs that were not changed or deleted. However, when the database contains a single area with fewer than 100,000 records, initializing the Index (using INIT) is generally faster than using REMOVE before performing a LOAD or Retrieve Index (RETIX). In all other cases, REMOVE is recommended.

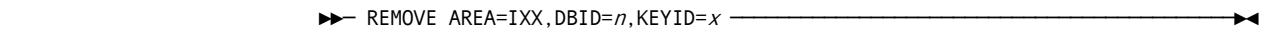
Changing a Key Definition Length

Use one of the following methods to rebuild a database Index after physically changing a key definition's length:

- Run the REMOVE and RETIX functions.
- Run the REMOVE and LOAD functions.
- Run the INIT and RETIX functions.
- Run the INIT and LOAD functions.

How to Use

Multi-User mode is required to execute this command. To perform the REMOVE function, supply the following command to DBUTLTY. The REMOVE function searches every index area for the database, so it works with the same options for single index area databases and Multi-Dataset Index areas.



Command

REMOVE

Invokes the function to remove a key ID from the Index Area.

Required Keywords

AREA=IXX

Identifies the area to be affected as the Index.

,DBID=

Specifies the database containing the key ID to be removed.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,KEYID=

Specifies the key ID to be removed. You should generate an Index Report to verify the key ID (see Index Area (IXX) Reports (see page 529)).

Valid Entries:

DATACOM-ID of the key in the database specified

Default Value:

(No default)

Example JCL

The following shows the command to remove Key ID 1 from the Index Area of database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          REMOVE    AREA=IXX,DBID=1,KEYID=1
 /*
 /&
 * $$ EOJ
```

Sample Report: REMOVE

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                 Copyright © 2009 CA. All rights reserved.                    *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REMOVE AREA=IXX,DBID=1,KEYID=1

                    FUNCTION=REMOVE
       ❷               AREA=IXX
                       DBID=1
                       KEYID=1

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX




                     IXX - STATUS INFORMATION
                   { DEVICE TYPE -    3380
                   { BLOCK LENGTH -  3,860
                   { BLOCKS/TRACK -     11
                   {
              ❹    { BLOCKS IN USE -
                   { BLOCKS UNUSED -        55
                   { TOTAL BLOCKS -         66
                   { TOTAL TRACKS -          6
                   { PERCENT FULL -         17
```

❹

Report indicating the status of the Index upon successful completion.

Chapter 26: REORG (Parallel BACKUP)

The parallel REORG function of the CA Datacom/DB Utility (DBUTLTY) provides a way for you to more quickly back up a large data area. By executing parallel tasks, the REORG function completes faster than a Native backup. To be parallel, the INNUM= keyword must be greater than one (1).

The result of the parallel REORG backup is 1-25 sequential files. These files must be used as input to a data LOAD (see LOAD (Build a Data Area) (see page 393)). When using more than one file, use the FILEIN function for each (see FILEIN (Concatenate Input) (see page 335)). Using FILEIN causes the files to be loaded by the LOAD function that follows.

For information about DBUTLTY STATUS console command support for this function, see DBUTLTY STATUS (see page 37).

For information about the impact of table partitioning on the REORG function, see The REORG Utility (see page 42).

REORG

When to Use

REORG is best used with areas that contain large amounts of data. For small areas, use the normal BACKUP and LOAD functions of the CA Datacom/DB Utility (DBUTLTY).

Note: If you are partitioning by some portion of Native Key, running a REORG DTFNAME=*,INNUM=*n* (where *n* is the number of partitions) gives the current way to break up the data set.

You can use the REORG function to do a parallel backing up of an area in Native Key sequence. A *reporting only* option is also available, if you use REORG OPTION2=BACKUPONLY and either DTFNAME=* or DTFNAME**. For information about DTFNAME=, see the list of Required Keywords.

Parallel Backup in REORG

The parallel backup portion of REORG allows you to use from 1 to 25 subtasks, each backing up a portion of a data area. Each subtask is assigned a key range and runs independently of other subtasks to get its assigned records to its sequential output file. If the data is slightly out of sequence, we recommend using only a few subtasks. In general, the more out of sequence the data, the more subtasks you should use. When the data is extremely out of sequence, if all buffers are not in memory, the backup process is I/O bound for large areas.

The processing flow for parallel backup using the REORG function is as follows:

1. Phase 1A: The area is opened for backup in DBUTLTY in a process similar to that of the BACKUP function. The processing sequence is always in native sequence, never in physical sequence.
2. Phase 1B: For each key ID that is a native sequence key for any table in the selected area, the low and high key range is selected for each backup subtask. If one task is selected, the key range is set as low values and high values. If more tasks are selected, the high level of the index is scanned to select a *best guess* at what the key ranges need to be, to approximate each subtask's reading of its portion of the data area. For example, if 4 tasks are requested, each should be assigned a key range to process about one fourth of the data rows.

   This process makes decisions based upon the high level of the index, and its effectiveness varies. The index is highly compressed, and the amount of data therefore represented by each entry varies. The goal is to read the highest level single block and use that information for the key selection. If that seems inadequate, the next lower level of the index is scanned. The default, 5 entries per task, is often a reasonable value. You can change this value, however, using the CYCLE= keyword, to cause more or less entries per task as required. This can be thought of as a trade-off between the effort it takes to select the key ranges, and the time when one subtask ends before the final subtask ends. The load cannot start until all backup subtasks are complete. Message **DB13259I REORG BACKUP KEYID-n LEVEL-n ENTRIES-n** provides details about the index selection, when more than one task is selected.

   It is possible and correct for one or more subtasks to be assigned no key values. This can occur for small areas or in the special case where a very large number of duplicate key values exist. For example, it could occur if an area had one million records and every record had a native sequence key of blanks, in which case the range that includes blanks is assigned to one of the tasks, and that task finds all the records. That task may be the first task, the last task, or a task in the middle. Other tasks run but quickly end without finding any rows, which is okay. The condition can also occur if tables in multiple areas share a native sequence key ID. In this case, one area has a very large number of records and another very few that are clustered in a limited key value distribution.

3. Phase 1C: Each subtask opens a task to the Multi-User Facility. Each executes a GSETL/GETIT-type request, with blocking, to find and get the data rows within its key range. If the data area and/or index are covered, the backup benefits. The blocking is at 32K, rounded down if the task size is smaller. Sufficient tasks must be available, or the normal return code 85 should be expected. The ACCESS setting for the database may *not* be OFF. The same is true for the ACCESS area.

At the completion of phases 1A, 1B, and 1C, a full and complete backup has been performed with the output data written to 1-25 sequential data sets. The utility function is now complete and the data sets are available for use.

If the index scanning and key range selection is severely deficient, the only solution is to do them manually using the FIRSTKEY/LASTKEY options of the BACKUP or EXTRACT functions.

The 1-25 sequential files produced by this backup process may be concatenated and provided as input to a regular LOAD function, either as all parallel parts concatenated together in key value order for a full load, or as less than all parts to allow for the deleting of a range of rows. The concatenation is done using the FILEIN function (see FILEIN (Concatenate Input) (see page 335)).

During the backup phase of REORG, you can request the status of the execution by issuing an Operating System *modify* command to the utility with a STATUS command. This presumes the DBSYSID macro producing a DBSIDPR program had the CONSOLE=YES option selected. If REORG is in the backup phase, this message is generated:

DB01323I - REORG BASE n AREA x RECORD y OF ABOUT z

In the message, the n is the DBID, the x is the area name, the y is the count of data records written to any of the output sequential files by any of the 1-25 sub-tasks running, and the z is the number of records in the area, as stored in the Directory (CXX).

Parallel Backup Restrictions

For a parallel backup, the area must be loaded in the URI format. The following restrictions apply to parallel backup:

- The area must be loaded in the URI format.
- For a full REORG, see the requirements for the LOAD function.

Parallel Data Reorganization

The parallel data load and parallel index update are requested by *not* specifying OPTION2=BACKUPONLY. They cannot be separated, and, if either fails, they must be restarted specifying OPTION2=LOADONLY.

The REORG function expects to process against a loaded data area with a loaded index. However, its ability to rerun from a previous error causes the function not to edit for these requirements.

The data load portion of REORG accepts the number of input files to be processed based upon input parameters. REORG starts a sub-task to process each input in parallel. Each of the subtasks opens a specific input file with a range of data rows (in native sequence) and adds them to the data area. The content of the data area at the start of the REORG (beyond the required data area control block) is not important. The content is completely replaced by the input data and empty space (low values). All of the subtasks execute simultaneously and/or concurrently, based upon processor availability.

Each subtask expects to load its share of the data, but because the amount of data within key ranges may vary, each subtask works with zero rows through all the rows in the area. If no input row is found by every subtask, an error is recognized and the parallel reorganization is aborted, that is, the REORG utility cannot be used to null load an area.

The parallel data reorganization is based upon the principle that the data needs to be in *better* native sequence key order, not in perfect native sequence key order. For perfect order, use the normal native sequence backup with a standard DBUTLTY LOAD function. The non-perfect ordering can be thought of as groups of rows ordered within the group by Native Key sequence.

Each subtask *knows* only the range of key values it is to read. It does not know how they compare to other subtasks. Also unknown is the number of rows each sub-task processes. Therefore, the basic flow of processing is for each subtask with a row(s) to add to acquire a logical group of tracks of the data area for its data. It adds its data in this group of tracks until it runs out of data or its group of tracks becomes full. If its group of tracks becomes full, the subtask gets another group of tracks. Because all the subtasks are working independently, rows within a group of tracks are in perfect order based upon the native sequence key values, but the Native Key sequence varies between groups.

We do not recommend using extracted files for input to the REORG utility for the following reasons:

- The table must have only unique native sequence keys. The native sequence key is defined with *unique,* or the native sequence key ID must be the same as the Master Key ID, and the Master Key must be defined as allowing no duplicate Master Keys.
- The REORG executes much slower, with additional I/O and CPU usage, when using extracted files than when using backup files. This is because, during the data reorganization, the key value must be extracted from each row and an index look-up done to discover the Unique Row Identifier (URI).
- If a duplicate is found or no index entry is found, the REORG utility stops immediately, leaving the data area unuseable, partly loaded with reorganized data, and partly with old data. At this point, the input data would have to be concatenated into the normal LOAD utility.

The default group is 15 tracks, but you can define it, by using the optional NUMBER= keyword, to be any desired number of tracks. A group of tracks acquired by a subtask for its use are not available to any other subtask. This requires that the data area must have sufficient free space, at completion of the data load, to allow a partial group for every subtask. Take this into account when selecting the number of subtasks. It is also required that at least one track be available for processing (not in a subtask group), once all the subtasks have completed.

The input data must be for an area loaded with URI=YES. The data area cannot use the DATASPACE option (DSOP=3) for clustering. The data area cannot contain a direct key table. If the input is in extract format, the data area being loaded must not contain a table defined as containing variable rows, or a table that allows duplicate native sequence key values. Extract format data also requires that the REORG build the key value from each row and perform an index look-up to find the URI of the rows.

As each row is placed in a data block, a memory table is updated with the block to which this row was, by URI, moved. The block in which the row originally occurred is not known and is not important. The memory table is stored in a series of data spaces. The number used is the minimum needed to contain the required information, based upon the highest URI in the area and the number of data blocks in the data area. Every URI number possible, from one through the highest currently in the area (as reported in the Directory (CXX) report), requires three or four bytes of space. Data areas of less than 16 million blocks require three bytes per URI. Larger data sets require 4 bytes per URI. The maximum number of data spaces that can be used is eight, but one or two are, typically, utilized.

The REORG utility expects to be executed with more than one subtask and to be provided multiple input files from multiple parallel backups or extracts, but it executes as a single task, with backup or extract input from a normal non-parallel backup or extract.

The parallel data load does *not* rebuild the free space index. At completion of the data load, the database can be opened for update. Any added rows go to the free space beyond the last assigned group of tracks. As soon as possible (next step), a DBUTLTY RETIX function should be executed with KEYNAME=*DATA, to rebuild the space index. The DBUTLTY RETIX function is a Multi-User Facility function. The database can be open and in the act of being updated during this process.

During the load phase, while any of the tasks still have data to load, the status of the execution can be requested by issuing an Operating System *modify* command to the utility with a STATUS command. This presumes the DBSYSID macro, producing a DBSIDPR program, had the CONSOLE=YES option selected. In the load phase, this message is generated:

DB01320I - REORG BASE n AREA x ON BLOCK y OF z RECORD r

In the message, n is the DBID, x is the area name, y is the highest block taken by any of the 1-25 load subtasks, z is the number of blocks initialized, and r is the number of records loaded by any of the 1-25 load subtasks.

Restrictions for Parallel Data Load and Parallel Index Update

The following are restrictions for parallel data load and parallel index update:

- This option is available only for a single area in a database at one time (MULTUSE=NO).
- The option is not available for clustered areas.
- The option is not available for areas containing a direct key table.
- The area may not contain variable length records.
- The option is not available for databases where any area in the database is not loaded in the URI format.
- The option is not available for areas without the index loaded or after an index has been initialized.
- The option requires current hardware supporting a full 2G data space (an error message indicates if not available).
- If using the LOADONLY option, we recommend you be extremely cautious in the specification and use of this function. Data rows could be lost if control statements are poorly defined, or due to poor data set handling, because of course it is not possible for CA Datacom/DB to edit the goals you desire to obtain by using the function. CA Datacom/DB can only edit the specific control statements that you provide.

- If using the LOADONLY option, backup format input cannot have been changed since the backup. Extract format input can have deleted rows only, not any additions or key value updates.

  Note: It is not possible to input new data into this utility that is not currently in the loaded index, or to have different key values than those that are in the currently loaded index. These factors cannot be edited.

- The area must be large enough to contain the input data and additional empty space, to allow the number of input files to each have a group of tracks reserved.
- This option is intended for single table areas or multiple table areas when all tables share one single native sequence key ID. However, it performs correctly with multiple tables and any (or all) not sharing a single native sequence key ID.
- The index update presumes a validly loaded index that is pointing to the data as it was at the time of the backup. It does *not* build an index, nor does it validate that the index matches the data. It can only know, for example, that URI 82 was in block 12 and is now in block 43, meaning that every index entry with URI 82 needs to have its block changed from 12 to 43.
- The internal URI (Unique Row Identifier) is *not* reassigned during the parallel reorganization. The URI has a maximum size of 4G. Beyond that value, no further additions to the data area are allowed, until the area is backed up and loaded with the standard LOAD function, which reassigns URI numbers starting with one. For large tables with many additions, this means that an occasional full BACKUP/LOAD is required.
- If the format is EXTRACT, then the table must have the native sequence key set to UNIQUE, or the native sequence key must be the same as the Master Key, and the Master Key must be set to allow no duplicate Master Keys. In addition, the rows cannot be defined as having variable lengths.

Parallel Index Update

The parallel index update follows the parallel data load in the same REORG function. The number of subtasks used to perform the parallel index update can range from 1 through 25. The default is 10. The physical tracks in the index data set are divided into the number of subtasks. Each subtask reads every block in its part, and updates every pointer as required. Blocks with a change are written. Blocks without a change are not written. Blocks are read and written as one or more tracks.

The parallel process first occurs for the index area IXX to completion. If the database has Multi-Dataset Index areas, the parallel process next occurs for each index area in turn.

The index update process is based upon the data received and the currently loaded index. It cannot validate or protect users from error conditions introduced by mishandling of data input files. For example, the utility does not sort input key values and compares them to the current index. If you were to add keys or change the fields of any current keys, missing or incorrectly indexed data would result. Be aware, however, that if a row or rows were deleted, as in a user program that purges data from the extract files, the utility does work, and you can take advantage of this fact, as required.

Post Parallel Reorganize Requirements

At this point the data has been loaded (in grouped Native Key sequence), and the index has been updated to reflect current data positions. However, no backup exists for future *image copy* restores that are needed for forward recovery processing. Therefore, for that purpose, at this point a new copy of the data area is necessary. A normal backup (SEQ=PHYSICAL) could be used for this purpose, or a standard DASD utility could be used.

Note: The parallel data reorganization places records in groups of tracks in a time dependent manner. A parallel reorganization is never directly repeatable, and therefore it cannot be used as a stable point for forward recovery.

The space index was deleted after the parallel index update. For any slack space before the end of the data to be found (for additions), it must be rebuilt. Do this using the RETIX function, with KEYNAME=*DATA.

BACKUPONLY or BACKUP phase of BACKUPLOADExecution Requirements/Controls

Environmental **Requirements** *when MULTUSE=YES/NO*

- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/READ/UTLTY.
- The area ACCESS must be set WRITE/READ/UTLTY.
- The area may have no table in an open URT with UPDATE=YES.
- The area may *not* have any other MULTUSE=YES utility executing.

Environmental **Controls** *when MULTUSE=YES/NO*

- The database will be opened by Multi-User Facility for update.
- The database will have no *unloading* status set.
- The utility sets ACCESS area READ (utility set).
- The utility sets REORG-BK executing this Multi-User Facility this database or area.

LOADONLY or LOAD phase of BACKUPLOADExecution Requirements/Controls

Environmental **Requirements** *when MULTUSE=NO*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any Multi-User Facility for read.
- The database may *not* have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO*

- The database will be opened for update.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when MULTUSE=YES*

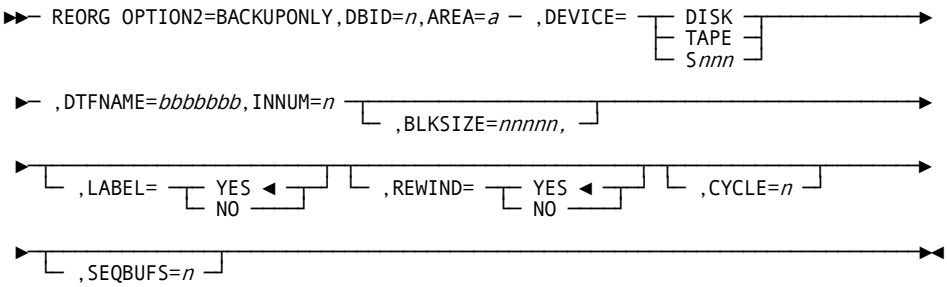
- The database may or may not be open for update in the connected Multi-User Facility
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The area may have no table in an open User Requirements Table.
- The area may *not* have any other MULTUSE=YES utility executing.
- The area must be closed.
- When MULTUSE=YES, this function will not be allowed against a Data Sharing Multi-User Facility if more than one is executing.

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets REORG-LD executing this Multi-User Facility this database or area.

How to Use

Multi-User mode is required to execute this command. Execute the REORG function using the following command format.



Command

REORG

Invokes the REORG function.

Required Keywords

OPTION2=

Specifies the type of REORG.

Valid Entries:

BACKUPONLY

Default Value:

(No default)

,DBID=

Identifies the database selected for REORG.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,AREA=

Identifies the area to be backed up.

Valid Entries:

DATACOM-NAME of the area in the database specified in the DBID= parameter

Default Value:

(No default)

,DEVICE=

When DEVICE=TAPE is specified, the system logical unit (SYS) number must be 10 (SYS010).

Use the S*nnn* entry to specify a specific SYS number (for tape only). For TAPE or S*nnn* this number is first the subtask then next sequential numbers are used for succeeding tasks. For example, with a specification of DEVICE=TAPE,INNUM=4 (where INNUM= specifies the number of subtasks to backup data), the output file SYS numbers that are expected are 10,11,12,13.

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

,DTFNAME=

Specifies the JCL TLBL/DLBL name for the output data set. If DEVICE=TAPE, this name must match the name which you code in your JCL TLBL statement; otherwise, this name must match the name you code in your JCL DLBL statement.

A *reporting only* option is available, if you use REORG OPTION2=BACKUPONLY and either DTFNAME=* or DTFNAME** as described in the following:

DTFNAME=*

Specifies that a report is to be generated, with the key values selected for each Native Key sequence ID, by providing starting and ending key values, in character and hexadecimal formats. If the starting and ending values are identical, no rows are found. The generated report can be used to assist in key selection if you prefer to back up the data using the FIRSTKEY/LASTKEY options of the BACKUP function.

DTFNAME=**

Specifies a report that can be used for debugging. It shows the full starting key to be assigned to each task, when running the REORG with a proper DTFNAME.

Valid Entries:

Any name valid to JCL specifications (1-5 characters, the number of the subtask based on INNUM= being suffixed as 1 through 25)

Note: There is also the special case of the use of * or ** (as previously described).

Default Value:

(No default)

,INNUM=

Specifies the number of subtasks to backup.

Note: If you are partitioning by some portion of Native Key, running a REORG DTFNAME=*,INNUM=*n* (where *n* is the number of partitions) gives the current way to break up the data set.

Valid Entries:

A number, 1 through 25

Default Value:

(No default)

Optional Keywords

,BLKSIZE=

Indicates the block size for the output data set. For best performance, use 32767 when going to tape and the largest block size which uses all of a track when going to disk.

Note: If the block size is not large enough to contain the largest record in the area (see the following formula), DBUTLTY terminates abnormally.

Calculate the minimum block size for the output data set as follows:

A + C + 30
A is the length of the largest record in the area,
C is 0, if the data is not compressed, or
C is (A/128) + 1, if the data is compressed.

Valid Entries:

A 1- to 5-digit number

Default Value:

4096 (for DEVICE=DISK), or
32767 (for DEVICE=TAPE)

,LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), specifies whether or not tape labels are to be processed. We recommend that labeled tapes always be used.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,CYCLE=

Specifies the minimum number of high-level index entries per task, to divide work. This specification is used during the backup phase of the REORG function.

Valid Entries:

2 through 99

Default Value:

5

,SEQBUFS=

Specifies the number of sequential buffers for each Multi-User Facility task that is backing up data. Even though allowed by the DBURTBL macro, in this case the value of SEQBUFS= may not be a minus number. To specify minus numbers, add 128. The default of 256 is equivalent to -128 (a negative 128). If you specify less than 8 for SEQBUFS=, the specification is rounded up to a value of 8.

Valid Entries:

0 through 256

Default Value:

256

,REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. NO allows the stacking of backups on one tape.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

Example JCL

Following is example JCL for a REORG BACKUPONLY.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,devicetype,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA
 // EXTENT SYSnnn,volser
 // DLBL PAY001,'db001.PAY',,DA
 // EXTENT SYSnnn,volser
 // DLBL BACK1,'backup1'                                    Ouput data
 // EXTENT SYSnnn,volser
 // DLBL BACK2,'backup2'                                    Output data
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
                     REORG OPTION2=BACKUPONLY,DBID=1,INNUM=2,
                           DTFNAME=BACK,DEVICE=DISK,AREA=PAY
 /*
 /&
 * $$ EOJ
```

Sample Report

Following is a REORG BACKUPONLY report example.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶           REORG OPTION2=BACKUPONLY,DBID=1,INNUM=2,DTFNAME=BACK,
                          DEVICE=DISK,AREA=PAY

                    FUNCTION=REORG
        ❷             AREA=PAY
                      DBID=00001
                      DEVICE=DISK
                      DTFNAME=BACK
                      INNUM=002
                      OPTION2=BACKUPONLY
        ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:   001         ******************************************************************************** Directory: PRODCXX
        ❹          TABLE          RECORDS
                    PAY              200
                    TOTAL            200
```

❹

Table and records information.

Chapter 27: REPLACE (Replace All Data in a Table)

REPLACE

The REPLACE function of DBUTLTY provides the ability to load one table in a multiple table area with new data, replacing all current data for that table. It executes in the Multi-User Facility and allows full read and update access to every other table in the same area and database to other applications in the same Multi-User Facility. For example, you could extract a table that is not a partitioned table, redefine it to be a partitioned table, and use the REPLACE function to populate it, provided all the other rules are met, such as the table to be replaced not being the only table in an area.

FILEIN may be used before a REPLACE. See FILEIN (Concatenate Input) (see page 335).

Restrictions

- The area must be loaded in the URI format.
- The area may not be clustered.
- The input data must be in EXTRACT format.
- The table may not have variable length records, for example in CA Datacom VSAM Transparency.
- The table may not be relative record.
- The table may not be the only table in the area.
- Sort is required.
- May not name a partition Full Parent or Any Parent.

When to Use

REPLACE is a special purpose function and is not a general replacement for the LOAD function. For reasons related to performance and ease of use, instead of placing every table in a database in its own area it is often better to place multiple small tables in either a single area or in a few areas. Although a single table in a multiple-table area cannot be loaded, the REPLACE function provides you with a similar ability. However, be aware that it is probably best to place a table with a large number of rows in its own area and a table with a very large number of rows in its own database. The area can be conveniently backed up and loaded as an area or a database. Occasionally, one table in a multi-table area needs to be replaced with no impact on other tables, such as when a table is being redefined. This specific niche is the reason for this function and, as such, it has numerous restrictions.

If the input file is empty, the table is effectively null loaded.

This utility function executes under control of a Multi-User Facility but can also contain Single User processing. The table cannot currently be open for the REPLACE function to start. When the REPLACE function starts, it marks the table as not loaded and, when successfully completed, it sets the table as loaded. The function opens the database for update in the Multi-User Facility and opens it for read in Single User processing.

The date/time loaded as reflected in the Directory (CXX) report and the Dynamic System Table DIR_TABLE is updated to reflect the REPLACE function. However, this assumes that the Directory is Version 11.0 style. Running in a compatibility mode with a r10 style, the date and time reflects the last LOAD function and is not updated for a REPLACE function.

Even though the data replace processing occurs in the Multi-User Facility, the deletion of old records and insertion of new records are not subject to logging. Therefore, no transaction backout can occur. If the utility fails, you must restart it.

If the area has insufficient space for the data or index, a dynamic extend is attempted. If it fails to find enough space, a return code 07 or 08 occurs. Perform a DBUTLTY EXTEND and restart the REPLACE function.

Note that new Unique Row Identifiers (URIs) are assigned to the added data. The numbers only start over from 1 when an area is loaded (when not using RECID=YES). If the high number should reach four billion, additions stop with a return code. In this case, you must back up the area and reload before more additions (including those with the REPLACE function) can be done.

The console status messages provided during a sorted LOAD or RETIX function also occur for the REPLACE function. Note however that, since much of the work being done by the REPLACE function occurs in the Multi-User Facility address space, the progress often occurs in 'spurts' and some progress information is not available.

REPLACE does not affect the MASSADD function. REPLACE takes the place of most MASSADDs, however, for those who use MASSADD to load multiple table areas.

Highlights of the Utility Side of the Processing

- Reads the input and blocks rows to MUF at the task area size.
- Builds index records for sorting.
- Directs sorting.
- Accepts sorted keys and sends blocks to MUF for index merge.

Highlights of the Multi-User Facility Processing

- Reads area in physical sequence with SEQBUFS=128.
- For each block read, deletes all rows for the table being replaced.
- If block has space and new data exists, adds new rows to the block.
- Space management updated as required for additions and deletions.
- If block changed, is scheduled for pipeline write.
- Assigns new URI to each row.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=YES*

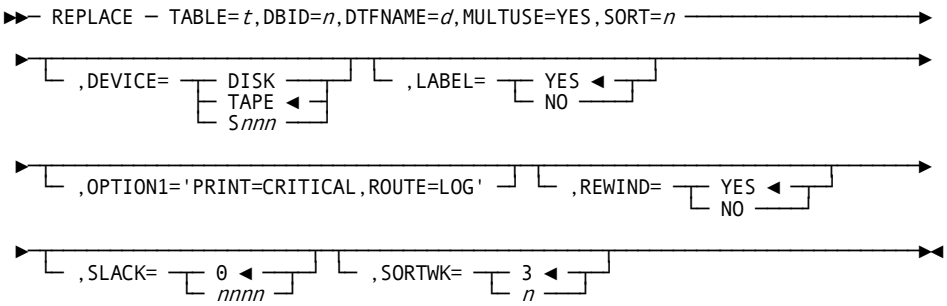
- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The table may not be in an open User Requirements Table.

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility ACCESS AREA is not set (table level function not area level).

How to Use

Multi-User mode is required to execute this command. Execute the REPLACE function using the following command format.



Command

REPLACE

Invokes the function to load one table in a multiple table area with new data, replacing all current data for that table.

Required Keywords

,DBID=

Identifies the database containing the table to be loaded.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,DEVICE=

Specifies the device type for the input data set.

When DEVICE=TAPE is specified, the system logical unit (SYS) number must be 5 (SYS005).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

,DTFNAME=

Specifies the z/VSE JCL TLBL/DLBL name for the input data set.

If DEVICE=TAPE, this name must match the name that you code in your z/VSE JCL TLBL statement. Otherwise, this name must match the name you code in your z/VSE JCL DLBL statement.

Valid Entries:

Any name valid to z/VSE JCL specifications

Default Value:

(No default)

,MULTUSE=YES

Specifying MULTUSE=YES is required when executing the REPLACE function.

Valid Entries:

YES

Default Value:

(No default)

,SORT=

Sorting index entries is required.

To estimate the value to enter, multiply the number of keys in the table by the number of records to be sorted and add 30 percent. For example, if the table contains 2 keys and you are sorting 1000 records, the value you specify should be at least 2600. The value you specify is passed to the sort routine and can affect the amount of runtime used by the sort. If you use multiple LOAD or REPLACE statements in a single step, place the largest sort value first.

Note: Do *not* specify over 99 million unless you verify that the sort package you are using allows a larger number.

Valid Entries:

0 to 99999999999

Default Value:

(No default)

TABLE=

Indicates a single table is to be loaded.

Valid Entries:

DATACOM-NAME of a table in the database and area being loaded

Default Value:

(No default)

Optional Keywords

,LABEL=

If DEVICE=TAPE (either TAPE or the S*nnn* form), specifies whether tape labels are processed. We recommend that you always use labeled tapes.

Note: LABEL= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

YES

,OPTION1=

Specifies the statistics display option to be passed to the sort program (see the SORT= parameter).

The OPTION1= parameter is highly recommended to obtain warning and critical messages if a problem occurs.

NOF

Corresponds to the NOFLAG parameter of the IBM sort. Only critical messages are to be displayed on the console.

'(I)'

Specifies that informational and critical messages are to be displayed on the printer and the console.

'(U)'

Specifies that only critical messages are to be directed to both the printer and the console.

NOF, '(I)', and '(U)' correspond to the Syncsort MSG options of CC, SC and CB respectively.

This parameter also accepts any three-character (including the surrounding parentheses) FLAG parameter that can be processed by the IBM sort. See your IBM sort manual for details.

Valid Entries:

NOF, '(I)', or '(U)'

Default Value:

NOF

,REWIND=

If DEVICE=TAPE (either TAPE or the S*nnn* form), YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. If omitted, the tape is rewound before the OPEN and rewound with unload after the CLOSE.

Note: REWIND= has no effect when DEVICE=DISK.

Valid Entries:

NO or YES

Default Value:

The tape is rewound before the OPEN and rewound with unload after the CLOSE.

,SLACK=

Specifies the number of bytes to reserve in each data block for future expansion of records in that block, or for addition of records when using a space reclamation option. The number entered applies only to blocks loaded to an area during this execution of the REPLACE function. Blocks not loaded with data use the SLACK value specified in the AREA definition in CA Datacom/DB.

Valid Entries:

A number that is less than the block size

Default Value:

0

,SORTWK=

Specifies the number of sort work areas.

Note: Do not specify a 2-digit number unless the sort package you are using allows a 2-digit number. Sort packages that do not support a 2-digit number can fail in an unpredictable manner, if you specify a 2-digit number.

Valid Entries:

0 to 99

Note: Do not use two-digit numbers unless the SORT package to be used supports the larger number.

Default Value:

3

Example JCL

Following is example JCL for the REPLACE function for table F01 in DBID 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL BACKUP,'input'                          Input data set
 // EXTENT ,volser
 // DLBL CXX,'cxx',,DA                           Directory data set
 // EXTENT ,volser
 // DLBL IXX001,'db001.index',,DA                Index for DBID 1
 // EXTENT ,volser
 // DLBL F01001,'db001.f01',,DA
 // EXTENT ,volser
 // ASSGN SYS001,device type,VOL=volser,SHR
 // ASSGN SYS002,device type,VOL=volser,SHR
 // DLBL SORTWK1
 // EXTENT SYS001,volser,,,rel trk/blk,number trks/blks
 // DLBL SORTWK2
 // EXTENT SYS002,volser,,,rel trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=768K
                 REPLACE DBID=1,TABLE=F01,DTFNAME=BACKUP,SORT=100,
                   SORTWK=2,OPTION1='PRINT=ALL',DEVICE=DISK,MULTUSE=YES

 /*
 /&
 * $$ EOJ
```

Sample Report: REPLACE

Following begins a REPLACE sample report.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                   Copyright © 2009 CA. All rights reserved.                    *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶              REPLACE DBID=1,TABLE=F01,DTFNAME=BACKUP,SORT=100,
                              SORTWK=2,OPTION1='PRINT=ALL',DEVICE=DISK,MULTUSE=YES

                    FUNCTION=REPLACE
        ❷              DBID=00001
                       DEVICE=DISK
                       DTFNAME=BACKUP
                       MULTUSE=YES
                       OPTION1=PRINT=ALL
                       SORT=000000100
                       SORTWK=02
                       TABLE=F01

        ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                   Copyright © 2009 CA. All rights reserved.                    *               SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX

        ❹            TABLE          RECORDS

                    F01                10
```

❹

Table and records information.

Chapter 28: REPORT (Generate Reports)

Overview

The REPORT function produces the reports in the preceding listings.

When to Use

You can generate reports at any time.

Length of Time to Execute REPORT

Each Statistics and Diagnostics Area (PXX) Report option that you request causes the entire Statistics and Diagnostics Area to be read. If your Statistics and Diagnostics Area data is allocated in a large area, the elapsed time to execute this function can be lengthy.

Directory (CXX) Reports

The REPORT function provides two types of reports about the information contained in the Directory.

- The Directory (CXX) Report shows the current status and control information for the Index, database, area, table, key, element, and special fields.
- The Directory (CXX) Report with TYPE=B shows the current status and control information for the index, database, area, and table. It excludes information for key, element, and special fields.
- The Space Utilization Report (selected by using the TYPE=A keyword) provides information about total tracks, records, blocks, and maximum percentages of block utilization and the space reclamation option specified.

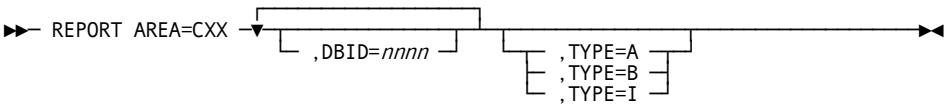
Note: You can obtain most of the Directory (CXX) information in real time through SQL access to special system tables. These dynamic tables allow easy access of the data from any platform while the Multi-User Facility is running. See the *CA Datacom/DB System Tables Reference Guide*.

When to Use

Use the information provided in a full Directory (CXX) Report to verify the current status and definitions currently in use. Use the space utilization option of the Directory Report to monitor space in your Directory, data, and Index areas. You can generate a Directory Report at any time. The Directory (CXX) reports read and reflect the information from the CXX data set. This information is correct when the database is not currently open by the Multi-User Facility or any Single User jobs, including DBUTLTY. Even when it is open, the information is usually adequately current. If opened by the Multi-User Facility, the information can be made *more* accurate by executing the COMM STATS option prior to the report, because executing COMM STATS refreshes the data in the data set. The utility does not interact with the Multi-User Facility. Although very unlikely, it is possible for a return code **04 INVALID RECORD ID,** with an internal return code of 47, to occur if maintenance is occurring at the same time the report is executing. Maintenance in this case includes the catalog of a database or SQL DDL processing. If this happens, the report should be executed again.

How to Use

You can execute this command in either Single User or Multi-User mode. Use the following control statement to request this report:



Command

REPORT

Invokes the function to produce a report.

Required Keywords

AREA=CXX

Requests the Directory Report.

Optional Keywords

,DBID=

DBID= is optional and may repeat. Each occurrence may specify one valid database ID or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250).

If DBID= is not specified, all databases are processed. That is, if a report is done without a DBID, the first page of the CXX report (not TYPE=A) has information about the full CXX.

If you use TYPE=I, you must also use a DBID= specification, that is, either one or more than one DBID, or a range of DBIDs. For TYPE=I the valid DBID specifications are from 1 through 5000.

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database in a range not defined is ignored.

Valid Entries:

One or more valid database IDs or a range of DBIDs as already described.

Default Value:

Omission of this keyword generates a report for all databases identified in the Directory.

,TYPE=A

Specifies the Space Utilization Report be generated. Omitting this parameter produces a full Directory report. For more information on the TYPE=A report, see Type A - Directory Space Utilization (see page 523).

,TYPE=B

Specifies a full report without key, element, and special fields. For more information on the TYPE=B report, see Type B - CXX Report (see page 527).

,TYPE=I

Specifies the Impact Report be generated. Omitting this parameter produces a full Directory report. For more information on the TYPE=I report, see Type I - Impact Report (see page 527).

Full Directory Report

To obtain current status and control information, generate a full Directory Report.

Example JCL

The following shows the command to generate a full Directory Report.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        REPORT   AREA=CXX
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT CXX - Full

If a report is done without a DBID, the first page of the CXX report (not TYPE=A) has information about the full CXX. In the following example, a full CXX Report report (without a DBID specified) displays the Version 12.0 enhancements in the report. It shows previous and new (for Version 12.0) field values by displaying the new field values on the right side of the example.

```
CONTROL AREA DIRECTORY
DBCRBAS REQUIREMENT (ALL BASES OPEN)  -    63,232  CXX ENQ - SYSPLEX
NUMBER OF DATA BASES -    34                     DATA HIGH USED MARK - NO
DEVICE TYPE - 3390                               DATASET EXTENT VALIDATION - NO
DYNAMIC FILE ALLOCATION ALLOWED - YES            DATA FAST SEARCH - NO
SECURE USING JOBNAME - NO                        CXX LEVEL -    1
SECURE ALLOWING SINGLE USER - YES                SINGLE USER ALLOWED - NO
SQL MODE - SUPPORT ACTIVE - YES
```

Following is a sample full CXX report showing partitioning for, in this example, a database with a DBID of 999.

Note: This example shows a few sample pages of the report. An actual report could of course be longer, continuing for as many pages as necessary.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT AREA=CXX,DBID=999

                    FUNCTION=REPORT
       ❷               AREA=CXX
                      DBID=00999

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:   999         ******************************************************************************** Directory: PRODCXX

DATA BASE INFORMATION

DBCRBAS REQUIREMENTS FOR THIS BASE -     2,392 } ❶
DBCRKEL REQUIREMENTS FOR THIS BASE -     4,128 }
NUMBER OF AREAS IN THIS BASE    -      3 }
NUMBER OF TABLES IN THIS BASE   -      9 } ❷
NUMBER OF KEYS IN THIS BASE     -     33 }
NUMBER OF ELEMENTS IN THIS BASE -      9 }
LAST OPENED     - DATE--5/08/2009, TIME-07.35.25 }
LAST RESTARTED  - DATE--5/08/2009, TIME-07.35.24 } ❸
LAST MAINTAINED - DATE--9/05/2008, TIME-06.42.18 }
REPLICATED BASE  - NO                                    BASE FORMAT -   2 } ❹
PARTITIONED BASE - NO                                                    }
UPDATE USERS    -   0 } ❺
READ ONLY USERS -   0 }
OCCURRENCE - TOBIE_DATABASE_PTNP001 ❻
SQL SECURITY - NO ❼

INDEX INFORMATION - IXX ❽
DEVICE TYPE  -   3390 } ❾                          DYNAMIC EXTEND -    YES
BLOCK LENGTH -  4,096 }                            DYN.EXT.TRACKS -     15
LEVELS      -      1 ❿                             OLDEP          -    YES
BLOCKS/TRACK -    12     }                         CXXMAINT LINK  -    YES
BLOCKS IN USE -        3 }
BLOCKS UNUSED -       57 }
TOTAL BLOCKS  -       60 }
TOTAL TRACKS  -        5 }
PERCENT FULL  -        5 }
DATA SET NAME - JBND007.IXX5000 }                  INDEX DTF/DDNAME - IXX5000
VOLUMES       - WRKD26          }
```

Database Information

❶

DBCRBAS reports the number of bytes required to hold the control information for this database in memory when it is open. For information only, DBCRKEL reports the number of bytes required to hold all the key and element definitions for this database.

❷

NUMBER OF ... IN THIS BASE is the count of the active areas, tables, keys, and elements in this database. If a key ID exists in more than one table, it is counted once for each table in which it occurs.

❸

These fields report activity involving this database:

LAST OPENED

Date and time of the last User Requirements Table open. If the table is currently open for read-only, the date and time can reflect a previous open for update. If the table was opened by multiple tasks concurrently, the date and time can reflect any of the tasks.

LAST RESTARTED

Date and time of the last RESET CXX for this database.

LAST MAINTAINED

Date and time that this database definition was last cataloged or maintained using the CXXMAINT options, except the ALTER option.

Note: Certain SQL DDL statements update the CXX (such as CREATE TABLE, CREATE INDEX, and so on) and modify this date. Since these occur internally, you may not realize that the CXX has been updated on the date and time reflected in this field.

❹

REPLICATED and PARTITIONED BASE indicate whether the database is replicated or partitioned. (See the *CA Datacom STAR Administrator Guide* for details on partitioning and replication.)

BASE FORMAT can be 1 or 2. FORMAT 1 restricts features of Version 12.0 that are not compatible with the previous release. FORMAT 2 allows all Version 12.0 functionality. For details about base format, see the information about upgrading the CXX in the chapter about creating and maintaining the Directory (CXX) in the *CA Datacom/DB Database and System Administration Guide*.

❺

UPDATE USERS reports the number of tables open for update times the number of User Requirements Tables open for update that contain the tables. (This count is not always maintained accurately on DASD.)

READ ONLY USERS reports the number of tables open for read only times the number of User Requirements Tables open for read-only that contain the tables.

❻

OCCURRENCE is the entity-occurrence name for the database, and the status/version in CA Datacom/DB.

❼

SQL SECURITY indicates whether SQL Security has been selected for the tables in this database. See the *CA Datacom/DB Database and System Administration Guide* for more information about security.

❽

The INDEX INFORMATION - IXX heading indicates the start of the index information for the IXX Index Area. The IXX is always the first (or only) index area. If this database has a Multi-Dataset Index, after the information for the IXX, information is provided about each additional index area.

❾

DEVICE TYPE and BLOCK LENGTH report the type of device containing this Index Area (IXX) and the block size chosen by CA Datacom/DB for this Index Area.

Note: When the Index Area or data area has not been INITed, the DEVICE is blanks.

❿

LEVELS is the count of the number of high-level Index levels. See the *CA Datacom/DB Database and System Administration Guide* for information about the CA Datacom/DB Index structure.

(page 2 continued)

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:   999         ******************************************************************************** Directory: PRODCXX

DATA BASE INFORMATION

DBCRBAS REQUIREMENTS FOR THIS BASE -     2,392 }
DBCRKEL REQUIREMENTS FOR THIS BASE -     4,128 }
NUMBER OF AREAS IN THIS BASE    -      3 }
NUMBER OF TABLES IN THIS BASE   -      9 }
NUMBER OF KEYS IN THIS BASE     -     33 }
NUMBER OF ELEMENTS IN THIS BASE -      9 }
LAST OPENED     - DATE--5/08/2009, TIME-07.35.25 }
LAST RESTARTED  - DATE--5/08/2009, TIME-07.35.24 }
LAST MAINTAINED - DATE--9/05/2008, TIME-06.42.18 }
REPLICATED BASE  - NO                                    BASE FORMAT -   2 }
PARTITIONED BASE - NO                                                    }
UPDATE USERS    -   0 }
READ ONLY USERS -   0 }
OCCURRENCE - TOBIE_DATABASE_PTNP001
SQL SECURITY - NO

INDEX INFORMATION - IXX
DEVICE TYPE  -   3390 }                          DYNAMIC EXTEND -    YES ❸
BLOCK LENGTH -  4,096 }                          DYN.EXT.TRACKS -     15 ❹
LEVELS      -      1                             OLDEP          -    YES ❺
BLOCKS/TRACK -    12     }                       CXXMAINT LINK  -    YES ❻
BLOCKS IN USE -        3 }
BLOCKS UNUSED -       57 }
TOTAL BLOCKS  -       60 } ❶
TOTAL TRACKS  -        5 }
PERCENT FULL  -        5 }
DATA SET NAME - JBND007.IXX5000 }                INDEX DTF/DDNAME - IXX5000 ❼
VOLUMES       - WRKD26          } ❷
```

❶

These lines report the space utilization for this Index Area:

BLOCKS /TRACK

Number of Index blocks per track.

For FBA devices, this is the number of physical blocks in one logical block.

BLOCKS IN USE

Number of Index blocks in use. Two blocks are always reserved for CA Datacom/DB control information. If a system failure has occurred during processing, this number can be inaccurate.

BLOCKS UNUSED

Count of the completely empty blocks in the Index. If a system failure has occurred during processing, this number can be inaccurate.

TOTAL BLOCKS

Total blocks used by the Index.

TOTAL TRACKS

Total tracks used by the Index.

PERCENT FULL

Number of used blocks divided by total blocks. The value reported is an even percent. Decimals are truncated. If a system failure has occurred during processing, this number can be inaccurate.

❷

DATA SET NAME is the data set name, if it is known. If the data set name is reported, a VOLUMES entry is included with volume serial numbers displayed. If necessary, multiple VOLUMES entries may be listed with 10 volume serial numbers per entry. The REPORT function obtains the volume serial numbers from CA Dynam/D for z/VSE if it is present and if the data set is catalogued.

❸

DYNAMIC EXTEND is specified as YES or NO for the Index Area.

❹

For DYN.EXT.TRACKS - *n*, the *n* provides that dynamic extend, whether set or not, is defined in tracks and not cylinders. If defined in cylinders, the field shows DYN.EXT.CYLS - *n*.

❺

OLDEP refers to *old entry points.* When OLDEP=NO is specified, the DBNTRY entry point is the only entry point supported. See the INIT function in the *CA Datacom/DB DBUTLTY Reference Guide* for information on the OLDEP= keyword.

❻

CXXMAINT LINK provides the current setting of the CXXMAINT option of LINK=YES/NO. Set to YES, the default and recommended option, the open of the area validates that the CXX name, in the data set opened, is the same as the CXX name in the executing CXX. This edit protects users from confusing production and test data sets. Set to YES, the default and recommended option, the open of the area will cause validation that the CXX name in the data set opened is the same as the CXX name in the executing Directory (CXX). This edit is to protect users from confusing production and test data sets.

❼

INDEX DTF/DDNAME - IXX5000 provides the DTF name or DD name that is used by CA Datacom to open this index area. The name is not an option.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:   999         ********************************************************************************* Directory: PRODCXX

TABLE NAME - K12  PARTITION - K01 2/3 }              AREA NAME  - P02                      }
OCCURRENCE - TOBIE_K12 P001           } ❶            OCCURRENCE - TOBIE_AREA_PTN_P02 P001   } ❷
TABLE ID   -    111                   }              AREA SQLNAME - TOBIE_AREA_PTN_P02      }
RECORD LENGTH    -     84 /     96  }                BLOCK LENGTH      -       4,096           SLACK -   0     }
RECORDS IN TABLE -               0  }                OVERFLOWS IN AREA -           0                          }
LENGTH OF KEY/ELEMENTS -  1,088     } ❸              BLOCKS IN USE -         2                                 }
NUMBER OF KEYS        -       9     }                BLOCKS UNUSED -        58                                 }
NUMBER OF ELEMENTS    -       1     }                TOTAL BLOCKS  -        60          CXXMAINT LINK  -    YES } ❹
CHANGE MASTER KEY    - YES }                         TOTAL TRACKS  -         5          DYNAMIC EXTEND -    YES }
DUPLICATE MASTER KEY - YES } ❺                       PERCENT FULL  -         3          DYN.EXT.TRACKS -     15 }
RECOVER  - YES } ❻                                   DEVICE TYPE  - 3390
LOGGING  - YES }                                     BLOCKS/TRACK -   12
PIPELINE - YES ❼                                     DSOP MODE    - NO RECLAMATION ❽
DB COMPRESSION   - NO } ❾                            TABLE LOADED - YES, DATE--5/08/2009, TIME-07.35.24 } ❿
USER COMPRESSION - NO }                              INDEX LOADED - YES                                 }
CLUSTER KEY ID     -   0 }                           DATADICTIONARY BASE  -     2
CLUSTER KEY LENGTH -   0 }                           DATA BASE ID CHANGE    -  NO
                         }                           DEFINITION POSTED VIA  - DATADICTIONARY CATALOG
AUTHID        - SYSUSR   }                           SQL-INTENT - Y                          }
TABLE SQLNAME - K12_PTN  }                           UNIQUE ROW IDENTIFIERS (URI) -         0 }
DOMAIN CONSTRAINT - NO }                             TABLE BEING REFERENCED       - NO  }
DL1-T CONSTRAINT  - NO }                             TABLE REFERENCING            - NO  }
TABLE CONSTRAINED - NO }                             ALL REFERENCING IN THIS BASE - YES }
CURRENT CHECK PENDING - LOAD SET          - NO }     HISTORY CHECK PENDING - LOAD SET          - NO }
                        RELATED TABLE     - NO }                           RELATED TABLE     - NO }
                        FORWARD RECOVERY  - NO }                           FORWARD RECOVERY  - NO }
TRIGGER(S)        - NO                        }      FORCED OFF        - NO }
CURRENT ROW(S) VIOLATE CONSTRAINT(S) - NO     }    HISTORY ROW(S) VIOLATE CONSTRAINT(S) - NO     }
DATA SET NAME -                 }                  DATA  DTF/DDNAME - P025000                    }
KEY - NAME   ID   LENGTH  MASTER  NATIVE  UNIQUE  DIRECT  KEY INC  CBS USE  OCCURRENCE  /  KEY SQLNAME           STATUS }
*     TST01  0001      8   YES     YES     NO       NO      YES      YES    TST01P001                            LD     }
                                                                            TOBIE_K01_PTN_TST01                          }
```

Table Information

❶

Identifies the table:

TABLE NAME

Name of a table in this database. If this is a table partition (Child) of a partitioned table, the table name represents the name of the partition. In addition, the literal PARTITION is present with 5 variable fields to assist in clarifying information about this partition. The first variable is the name of the Full Parent table. The next two variables, separated by a slash (/), are the relative number of this partition and the number of partitions. In this example, K12 is the second of three partitions, based upon the high value of this partition. The next two variables, separated by a dash (-), are the table name of the previous partition and the next partition, based upon the high values of the partitions. The first partition has blanks for the previous partition, and the last partition has blanks for the next partition.

OCCURRENCE

CA Datacom/DB entity-occurrence name for the table and its status/version.

TABLE ID

Internal table identifier.

❷

Identifies the area containing the previously identified table:

AREA NAME

DATACOM-NAME of the area. If multiple tables exist in this area, this information is duplicated for each table. If *HISTORY* follows the name, this entry has been logically deleted but still exists to provide the ability to back up the area.

OCCURRENCE

CA Datacom/DB entity-occurrence name for the area and its status/version.

AREA SQLNAME

Blank if table is not accessible by SQL, or if the database has not been cataloged with r9 or later.

❸

Reports information about the contents of this table:

RECORD LENGTH

The first value is the length of the user data which is delimited from the second value by a slash. The second denotes the length of the user data plus the CA Datacom/DB Record Control Element (RCE).

RECORDS IN TABLE

Number of records in the table. If a system failure has occurred during add/delete processing, this number can be inaccurate.

LENGTH OF KEY/ ELEMENTS

Amount of storage needed by the Multi-User Facility to hold the CXX control blocks that contain each key, element, or special field definitions for the given table.

NUMBER OF KEYS

Number of keys defined for this table.

NUMBER OF ELEMENTS

Number of elements defined for this table.

❹

Reports physical details about this data area:

BLOCK LENGTH

Length of the data blocks in this area. If the area is to have a new block size during the next data INIT, the current size prints followed by a slash and the new block size.

SLACK - *n*

The *n* specifies the slack size defined for the table in its definition.

OVERFLOWS IN AREA

Number of records added to the area since the last area load.

BLOCKS IN USE

Number of blocks in the data area that currently contain data. If a system failure has occurred during processing, this number can be inaccurate.

BLOCKS UNUSED

Number of blocks that currently are completely empty. If a system failure has occurred during processing, this number can be inaccurate.

TOTAL BLOCKS

Total blocks in this area.

CXXMAINT LINK - YES/NO

Provides the current setting of the CXXMAINT option of LINK=YES/NO. Set to YES, the default and recommended option, the open of the area validates that the CXX name, in the data set opened, is the same as the CXX name in the executing CXX. This edit protects users from confusing production and test data sets.

TOTAL TRACKS

Total tracks allocated to the area.

PERCENT FULL

Percent full is calculated by dividing blocks in use by total blocks. The value reported is an even percent. Decimals are truncated. If a system failure has occurred during processing, this number can be inaccurate.

DEVICE TYPE

Type of device containing this area.

Note: When the Index Area or data area has not been INITed, the DEVICE is blanks.

BLOCKS/ TRACK

Data blocks per track in this area.

DYNAMIC EXTEND

Specifies whether dynamic extends are in effect for this data area.

DYN.EXT.TRACKS - *n*

For DYN.EXT.TRACKS - *n*, the *n* provides that dynamic extend, whether set or not, is defined in tracks and not cylinders. If defined in cylinders, the field shows DYN.EXT.CYLS - *n*.

❺

CHANGE MASTER KEY and DUPLICATE MASTER KEY reports whether the Master Key value can be changed during update processing and whether duplicate Master Key values are allowed.

❻

RECOVER and LOGGING reports whether CA Datacom/DB recovery is activated for this table and whether logging is active for this table.

❻

PIPELINE indicates whether the Pipeline maintenance option is active for this table.

❽

DSOP MODE reports the data area space reclamation option for this table.

❾

DB COMPRESSION and USER COMPRESSION indicates whether CA Datacom/DB compression is active for this table and whether user compression is used with this table. If user compression is used, the name of the compression module is displayed.

❿

These fields indicate whether the table and index are loaded.

TABLE LOADED

- **NO**

  Specifies that the table is not loaded. When TABLE LOADED is NO and it was set not loaded as part of a catalog, then it may be followed by an explanation that might be one of the following, some of which involve information about partitioning:

  AREA NAME CHANGED The area for this table has changed, so the table has been marked as not loaded.

  ANOTHER PARTITION CHANGED The high-value value for the partition previous to this one has changed. Therefore, this partition must also be reloaded.

  COMPRESSION CHANGED The compression option has changed, so the table needs to be reloaded.

  COMPRESSION NAME CHANGED The name of user compression has changed, so the table needs to be reloaded.

  PARTITION COLUMNS UNEQUAL The number of columns in the partition key are unequal, so the table needs to be reloaded.

  PARTITION HIGH VALUES CHANGED The high value has changed, forcing the need to reload the table.

  RECOVERY CHANGE The type of recovery has changed, forcing the need to reload the table.

  RECORD LENGTH CHANGE The record length of the table has changed, forcing the need to reload the table.

  SPECIAL FIELD CHANGE The change in the number of special fields has changed, forcing the need to reload the table.

TABLE ID CHANGE The table ID has changed, forcing the need to reload the table.

TO/FROM PARTITION Either the new table is partitioned and the old table is not, or the old table is partitioned and the new table is not.

- **YES**

  Specifies that the table is loaded and the date and time of the last load are displayed.

INDEX LOADED

- **NO**

  Indicates that the index is not loaded. This condition exists after a new definition is cataloged which defines a database that did not exist before. Also, if the index is initialized, it would be considered not loaded. Other conditions like errors during a load may also set index not loaded.

  URI Indicates that the index exists for any required URI indexes and the space index is loaded. The data is not, however, indexed. This is normally set when a load is done with the INDEX=NO option. It can be set if a catalog is done which adds, deletes, or updates key definitions, so that the data is not considered indexed and must be rebuilt. The URI and space index, however, are correctly loaded. The forward recovery utility requires a YES or URI status to be able to correctly run.

- **YES**

  Indicates that the index is loaded with all data indexed.

(page 3 continued)

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:   999         ******************************************************************************** Directory: PRODCXX

TABLE NAME - K12  PARTITION - K01 2/3 }              AREA NAME  - P02                      }
OCCURRENCE - TOBIE_K12 P001           }              OCCURRENCE - TOBIE_AREA_PTN_P02 P001  }
TABLE ID   -    111                   }              AREA SQLNAME - TOBIE_AREA_PTN_P02     }
RECORD LENGTH    -     84 /     96  }                BLOCK LENGTH      -       4,096           SLACK -   0     }
RECORDS IN TABLE -               0  }                OVERFLOWS IN AREA -           0                           }
LENGTH OF KEY/ELEMENTS -  1,088     }                BLOCKS IN USE -         2                                 }
NUMBER OF KEYS        -       9     }                BLOCKS UNUSED -        58                                 }
NUMBER OF ELEMENTS    -       1     }                TOTAL BLOCKS  -        60          CXXMAINT LINK  -    YES }
CHANGE MASTER KEY    - YES }                         TOTAL TRACKS  -         5          DYNAMIC EXTEND -    YES }
DUPLICATE MASTER KEY - YES }                         PERCENT FULL  -         3          DYN.EXT.TRACKS -     15 }
RECOVER  - YES }                                     DEVICE TYPE  - 3390
LOGGING  - YES }                                     BLOCKS/TRACK -   12
PIPELINE - YES                                       DSOP MODE    - NO RECLAMATION
DB COMPRESSION   - NO }                              TABLE LOADED - YES, DATE--5/08/2009, TIME-07.35.24 }
USER COMPRESSION - NO }                              INDEX LOADED - YES                                  }
CLUSTER KEY ID     -   0 }                          DATADICTIONARY BASE  -     2 ❷
CLUSTER KEY LENGTH -   0 } ❶                         DATA BASE ID CHANGE    -  NO   ❸
                         }                           DEFINITION POSTED VIA  - DATADICTIONARY CATALOG ❹
AUTHID        - SYSUSR   }                           SQL-INTENT - Y                          } ❺
TABLE SQLNAME - K12_PTN  }                           UNIQUE ROW IDENTIFIERS (URI) -         0 }
DOMAIN CONSTRAINT - NO }                             TABLE BEING REFERENCED       - NO  }
DL1-T CONSTRAINT  - NO } ❻                           TABLE REFERENCING            - NO  } ❼
TABLE CONSTRAINED - NO }                             ALL REFERENCING IN THIS BASE - YES }
CURRENT CHECK PENDING - LOAD SET          - NO }     HISTORY CHECK PENDING - LOAD SET          - NO }
                        RELATED TABLE     - NO }                           RELATED TABLE     - NO } ❾
                        FORWARD RECOVERY  - NO } ❽                          FORWARD RECOVERY  - NO }
TRIGGER(S)        - NO                         }     FORCED OFF        - NO }
CURRENT ROW(S) VIOLATE CONSTRAINT(S) - NO      }     HISTORY ROW(S) VIOLATE CONSTRAINT(S) - NO      }
DATA SET NAME -                 } ❿                DATA  DTF/DDNAME - P025000                    }
KEY - NAME   ID   LENGTH  MASTER  NATIVE  UNIQUE  DIRECT  KEY INC  CBS USE  OCCURRENCE  /  KEY SQLNAME            STATUS }
*     TST01  0001      8   YES     YES     NO       NO      YES      YES    TST01P001                              LD     }
                                                                            TOBIE_K01_PTN_TST01                           }
```

❶

CLUSTER KEY ID and CLUSTER KEY LENGTH reports the cluster key ID and length for this table. (If clustering by table, the cluster key ID is 0 and length is 3.)

AUTHID and TABLE SQLNAME contain blanks, if not yet cataloged with r9 or later. If the table is accessible by SQL, in CA Datacom/DB the first and each r9 or later catalog provides information. If cataloged with r9 or later and the table is not accessible by SQL, AUTHID is blank and TABLE SQLNAME contains **'NO SQL ACCESS.'**

❷

DATADICTIONARY BASE the database ID of the CA Datacom/DB where the definitions were stored when this table was cataloged to the Directory.

❸

DATA BASE ID CHANGE indicates if the DBID has been changed during a LOAD AREA=CXX using NEWDBID=, or a CXXMAINT OPTION=DDPROD using NEWDBID=.

❹

DEFINITION POSTED VIA indicates if the table was posted to the Directory using the CA Datacom/DB CATALOG, the (DDCFBLD -CXX BUILD transaction and the CXXMAINT DDPROD option), or LOAD FROM BACKUP.

❺

SQL INTENT is Y if the table has been defined with full SQL access, R if the table is SQL read-only, or N if no SQL access is allowed, or * (an asterisk) if unknown because the structure was cataloged before the SQL INTENT option was supported.

UNIQUE ROW IDENTIFIERS (URI) indicates whether the area is being processed with Unique Row Identifiers (URIs). NONE indicates the area is *not* being processed with URIs. If the area *is* being processed with URIs, the highest URI assigned is displayed.

❻

These fields indicate which type of constraints apply for this table.

DOMAIN CONSTRAINT

Indicates that CA Datacom/DB domain constraints are in effect for this table.

DL1-T CONSTRAINT

Indicates that CA Datacom DL1 Transparency constraints are in effect for this table. See the *CA Datacom DL1 Transparency User Guide* for more information.

TABLE CONSTRAINED

Indicates if the table is constrained by SQL.

❼

These fields provide referential integrity information for this table.

TABLE BEING REFERENCED

Indicates whether this table is referenced by another table or is the object in a relationship with another table.

TABLE REFERENCING

Indicates whether this table references another table or is subject in a relationship with another table.

ALL REFERENCING IN THIS BASE

Indicates whether all relationships to this table exist in this database.

Note: If there is no relationship in either this database or any other, YES is indicated.

❽

Indicates whether the table needs to have referential integrity confirmed using the DBUTLTY CONFIRM function. The categories listed on the report indicate the function that was performed when the check pending was issued. See following item.

❾

Indicates the history of the table regarding referential integrity.

LOAD SET
RELATED TABLE
FORWARD RECOVERY

Indicates the function that was being performed when the check pending was issued.

FORCED OFF

Indicates whether a DBUTLTY CONFIRM function was executed using the FORCE parameter.

HISTORY ROW(S) VIOLATE CONSTRAINT(S)

Indicates whether data in the row or rows violated an integrity constraint or constraints.

DATA DTF/DDNAME - P025000

Provides the DTF name or DD name that is used by CA Datacom to open this data area. The name is not an option.

❿

The data set name, if it is known.

At the bottom of this report are all keys and their pertinent attributes defined for this table.

KEY SQLNAME

Is blank if the table is not accessible by SQL, or if the database has not been cataloged with r9 or later. Otherwise, it contains the SQL name of the key.

STATUS

STATUS on the KEY line occurs only if the base format is FORMAT 2. FORMAT 1 databases do not have a status at the key level. Below STATUS can be LD for loaded or NL for not loaded.

The asterisk (*) below the key is blanks if the database is FORMAT 1 and is blanks if the key described is to be stored in the IXX Index Area (the default). If this key is to have its values stored in a different index area of a Multi-Dataset Index, the asterisk is replaced by the name of the area, for example, I01.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     4
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:   999         ******************************************************************************** Directory: PRODCXX

TABLE K12 CONTINUED

    - FIELDS - ORDER    SENSITIVE  NULL  START  LENGTH  OCCURRENCE           }
                                                                             } ❶
               ASCEND   NOT SENS   NO       28       8  FIRST-RANDOM-NO P002 }
                 .
                 .
                 .

ELEMENTS - NAME   SECURITY CODE   START  LENGTH   DATA TYPE  NULL COL.  OCCURRENCE }  ❷
           TSTEL     00-000          0      84   CHARACTER     YES     TSTEL P002 }
```

❶

Report of all columns (fields) defined for this key.

ORDER

Indicates the sequence (ascending or descending) of field data in the index.

SENSITIVE

Indicates if the field is data type sensitive.

- FLOAT Floating-point number
- NOT SENS Not data type sensitive
- PACKED Packed decimal
- SIGN INT Signed integer
- ZONE DEC Zoned decimal
- NULL Indicates whether the field accepts null values.
- START Contains the displacement of the field in the table.
- LENGTH Contains the length of the field.
- OCCURRENCE Contains the entity-occurrence name and status/version of the field.

❷

Report of all elements and their pertinent attributes defined for this table. The element security code (if any) is displayed in both two-digit hexadecimal and three-digit decimal value format.

The element data type is always CHARACTER. The element null column (NULL COL) displays YES when the element has one or more fields whose columns can contain null values. NO indicates that the element has no fields that can contain null values.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     5
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:   999         ******************************************************************************** Directory: PRODCXX

TABLE K12 CONTINUED

PARTITION FULL PARENT NAMES  - K01 (SYSUSR                        ) (K01                          )  }
PARTITION -ANY- PARENT NAMES - KA1 (SYSUSR                        ) (KA1                          )  }
MOVE TO NEW PARTITION - NO   RESTRICTED  - YES                                                       }
PARTITION FIELD NAME             SENSITIVE  START  LENGTH   HIGH VALUE                               }
RECORD-NO                        ZONE DEC       4       8   15                                      }
DESCENDING-NO                    NOT SENS      44       8   99999991                                }
```

Partitioning Information

Table partitioning information.

PARTITION FULL...

Shows that K01 is the name of the Full Parent table in this database, in this example.

PARTITION -ANY-...

Shows that KA1 is the name of the Any Parent table in this database, in this example.

(SYSUSR)

Shows that SYSUSR is the table SQL authorization ID for both the Full Parent and the Any Parent table, in this example.

(K01)

Shows that the table SQL name of the Full Parent table is K01, in this example.

(KA1)

Shows that the table SQL name of the Any Parent table is KA1, in this example.

MOVE TO NEW PARTITION

Indicates whether an update is allowed to move to a new partition. YES indicates that if a partition key is updated and that update would cause the record to move to a new partition, the move will take place. NO means the move will not take place, and a return code will be received.

RESTRICTED

Refers to a restricted table, YES or NO. Applications cannot access Child tables *directly* within a partition if RESTRICTED - YES is indicated.

Note: Child tables can, however, in cases where RESTRICTED is YES, still be accessed through the Parent table or Any Parent table.

PARTITION FIELD NAME

Indicates the occurrence name of the field making up the partition key.

SENSITIVE

Indicates whether the field is data type sensitive.

- NOT SENS Not data type sensitive
- PACKED Packed decimal
- SIGN INT Signed integer
- ZONE DEC Zoned decimal

START

Shows the starting position of this field in the data record.

LENGTH

Shows the length of the field in the data record.

HIGH VALUE

Shows the high value of this partition. If this field contains unprintable characters, it will also be printed in hexadecimal format.

Sample Report: REPORT CXX - Full - Special Fields

A list of certain special fields can appear after the elements section on the report. To appear on the list, the field must have the DBEDITS or NULL-INDICATOR attribute turned on. The DBEDITS attribute is turned on for all columns (fields) when the table is created through SQL.

When the DBEDITS attribute-value for a FIELD entity-occurrence is set to Y, additional information is stored in the Directory during a catalog and the records are edited by CA Datacom/DB according to data type when they are added, loaded, or updated.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:    16         ******************************************************************************** Directory: PRODCXX

TABLE A11 CONTINUED

ELEMENTS - NAME   SECURITY CODE   START  LENGTH   DATA TYPE  NULL COL.  OCCURRENCE

           SQLEL     00-000          0      27   CHARACTER     YES     SQLEL P001


FIELDS - TYPE   NULL  FORCE  CHANGE  DEFAULT  VALUE               START  LENGTH  OCCURRENCE    }
         CHAR   NO    NO      YES    LITERAL  DEMOCHAR              0       8  DEMCHAR P001  }
         BIN    YES   NO      YES    LITERAL  9999                  8       5  DEMINT P001   }
         BIN    YES   NO      YES    NO DFLT                       13       5  DEMINT2 P001  }
         CHAR   YES   NO      YES    NO DFLT                       18       9  DEMCHAR2 P001 }
```

Report of the special columns (fields) for this table. Fields only appear if they have DBEDITS=Y or NULL-INDICATOR=Y.

TYPE

Indicates the data type of the field or, if it was placed on the list because NULL-INDICATOR=Y, NOEDT to signify that the DBEDITS attribute is not turned on for this field.

BIN Binary in a database cataloged in a previous version (see SBIN and UBIN)

BIT Character field, bit data (not converted to or from PC)

CHAR Character

DATE SQL DATE

FLOAT Floating-point

GRAF Double-byte character set (replaces DBCS)

MIXED Character field which allows single-byte and double-byte characters (double-byte characters must be within Shift-Out/Shift-In pairs)

NOEDT No DBEDITS on this field

SBIN Signed binary (see BIN and UBIN)

SPACK Signed packed decimal

STAMP SQL TIMESTAMP

SZONE Signed zoned decimal

TIME SQL TIME

UBIN Unsigned binary (see BIN and SBIN)

UPACK Unsigned packed decimal

UZONE Unsigned zoned decimal

VBIT Variable length BIT

VCHAR Variable-length character

VGRAF Variable length GRAF (double-byte character set)

VMIX Variable length MIXED

NULL

Indicates whether the field accepts no values (NULL-INDICATOR).

FORCE

Indicates whether a value must be provided.

ADD Must supply a value when adding (FORCEADD).

ALL Must supply a value when adding or updating (FORCEADD and FORCEUPD).

NO A value is not required when adding or updating.

UPDT Must supply a value when updating (FORCEUPD).

CHANGE

Indicates whether a change is allowed (UNCHANGEABLE).

DEFAULT

Indicates the type of default value.

LITERAL A specified value

NO DFLT No default

STD Standard default for type

USER The system user ID (AUTHID)

VALUE

Contains the specified value for a literal value.

START

Contains the displacement of the field in the table.

LENGTH

Contains the length of the field.

OCCURRENCE

Contains the entity-occurrence name and status/version of the field.

Type A - Directory Space Utilization

To obtain information about the number of tracks and blocks allocated and the number of blocks in use, and other important space utilization information, generate a Space Utilization Report. The keyword and value TYPE=A in the control statement cause this information to print.

Example JCL

The following shows the command to generate a Space Utilization Report for the Directory.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
         REPORT   AREA=CXX,TYPE=A
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT CXX - Space Utilization

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT  AREA=CXX,TYPE=A

                    FUNCTION=REPORT
       ❷               AREA=CXX
                       TYPE=A


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

                             DATA AREA SPACE UTILIZATION REPORT


   ❶            ❷           ❸          ❹          ❺        ❻          ❼         ❽

AREA DATA       TOTAL       TOTAL      TOTAL       USED    PERCENT     PARTIALLY  AREA REUSE
NAME BASE      TRACKS      RECORDS     BLOCKS     BLOCKS  FULL    MAX  EMPTY BLKS    OPTION

CXX              300         N/A      3,000        204     6      6         N/A  N/A

IXX       1        4         N/A         44         14    31     31         N/A  N/A
 DEM      1        2          17         62         11    17     17          80  CLUSTERING
 PAY      1        2         200         62         12    19     19           0  NO RECLAMATION
 PMF      1        2         200         62         16    25     25           0  NO RECLAMATION

IXX       2      140         N/A      1,540        817    53     53         N/A  N/A
 ALS      2        7         894         70         32    45     45           0  RANDOM
 ARA      2        1         106         10          5    50     50           1  RANDOM
 ATZ      2        5          22         55          2     3      3          19  RANDOM
 BAS      2        1          17         11          2    18     18           1  RANDOM
 DVW      2        1           6         11          2    18     18           1  RANDOM
 ELM      2        4         411         40         21    52     52           1  RANDOM
 FIL      2        2         183         22         10    45     45           0  RANDOM
 FLD      2       45       7,868        450        427    94     94           0  RANDOM
 HSD      2        6         192         60         37    61     61           0  RANDOM
  .    .           .           .          .          .     .      .           .    .
  .    .           .           .          .          .     .      .           .    .
  .    .           .           .          .          .     .      .           .    .
```

❶

AREA NAME is the DATACOM-NAME of the area and DATA BASE is the DATACOM-ID of the database containing the area. Summary statistics for the Directory (CXX) are always displayed on the first line.

❷

TOTAL TRACKS is the number of tracks allocated and accepted by CA Datacom/DB for use in the area. For FBA devices, this is the number of physical blocks allocated. If the actual number of tracks (blocks) allocated is larger, you should execute an EXTEND function to make the space available to CA Datacom/DB.

❸

TOTAL RECORDS - For data areas, the number of records in the area. If a system failure has occurred during maintenance processing, this number can be inaccurate.

❹

TOTAL BLOCKS - Number of blocks in the area.

❺

USED BLOCKS - Number of blocks that contain data. If a system failure has occurred during maintenance processing, this number can be inaccurate.

❻

PERCENT FULL and MAX - Percentage full is calculated by dividing used blocks by total blocks. The value reported is an even percent. Decimals are truncated.

Maximum percentage, sometimes referred to as "high-water mark," is calculated by dividing maximum used blocks by total blocks. Maximum used blocks is computed internally. This percentage is higher than FULL percentage when:

- All records are deleted in a data block with space management option 1 or 3 (random or cluster). See the *CA Datacom/DB Database and System Administration Guide* for information about the data area space management options.
- All Index entries are deleted in an Index block.

❼

PARTIALLY EMPTY BLKS - Blocks that contain sufficient space to hold the largest record in the area are considered to be partially empty. Even when all blocks contain data (percent full is 100), records can still be added, if partially empty blocks exist and a space reclamation option has been selected.

❽

AREA REUSE OPTION - The data area space management option used for the data area.

Type B - CXX Report

TYPE=B is equivalent to the full report for the database, area, and table information, but it does not report any key, element, or field information.

An estimated half of the printed lines in (and time spent printing) a full CXX report are used to give static information about keys, elements, and fields. Most CXX reports, however, are requested in order to view dynamic data information, in particular the data set names and volumes, and the table loaded status, all of which are available, without the static information, in a TYPE=B report.

Type I - Impact Report

Use TYPE=I when you want to request an Impact Report after a CA Datacom/DB CATALOG. An Impact Report shows you the impact of the CATALOG. The valid values for the DBID specified in a TYPE=I request can range from 1 through 5000.

The MUF must be enabled to use TYPE=I.

When an Impact Report is obtained after a CXXMAINT DDPROD or using TYPE=I after a CA Datacom/DB CATALOG, the reasons are printed on the report.

Example JCL

The following shows the command to generate an Impact Report for the Directory, database 16.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        REPORT AREA=CXX,TYPE=I,DBID=16
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT CXX - Impact Report

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT AREA=CXX,TYPE=I,DBID=16

                    FUNCTION=REPORT
       ❷               AREA=CXX
                       TYPE=I
                       DBID=16


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX

                                    IMPACT REPORT
  ❹           ❺           ❻    ❼
  BASE        AREA        TABLE ID        DISPOSITION
                                             ❽        ❾
  997         A01         C01  001        LOADED      (01)
                          C02  002        LOADED      (00)
                          F01  103        LOADED      (01)
                          F02  104        INDEX NOT LOADED (00)
                                                      KEY TST01 LOADED
                                                      KEY TST02 NOT LOADED
                          F03  105        LOADED      (01)
                          K01  240        LOADED      (00)
                          RNA  207        LOADED      (00)
                          RNB  208        LOADED      (01)
```

❹

The DATACOM-ID of the database containing the area.

❺

The DATACOM-NAME of the area.

❻

The table name.

❼

The table ID.

❽

The reason as listed in the table on page Processing a DDPROD Request (see page 247).

The TYPE=I report has been enhanced to provide information about cases in which the disposition (as shown in the previously shown example) of a table is INDEX NOT LOADED. In those cases, further information is provided regarding key-level status, that is, whether each individual key in the table is LOADED or NOT LOADED. This Impact Report enhancement allows you to see, after a catalog, what actions occurred relating to the need to load tables. With Version 12.0, a major feature is the index-level loaded status. In the previously shown example, the key-level status is listed for two keys, TST01 and TST02, in a table (F02 104) whose disposition is INDEX NOT LOADED. The rest of the Impact Report is as it was in the previous release, that is, tables whose disposition is LOADED have no need to also show the Version 12.0 key-level status.

❾

The code as listed in the table on page Processing a DDPROD Request (see page 247).

Index Area (IXX) Reports

The Index Area (IXX) Report provides information about keys. Specifically, it provides:

- The number of keys indexed for each table for the selected Index Area (IXX). You can request this report in either table name (TYPE=A) or key ID (TYPE=B) sequence.
- A dump of the Index Area associated with a specific key (TYPE=D) or a dump of the high-level portion of the Index Area associated with a specific key (TYPE=E). Use at Technical Support's request only.
- The number of records that have unique values for key field(s) (TYPE=G).

Note: TYPE=A,B,D,E: these have been changed from running Single User to running Multi-User. This provides current data, including the ability to inspect virtual database data.

When to Use

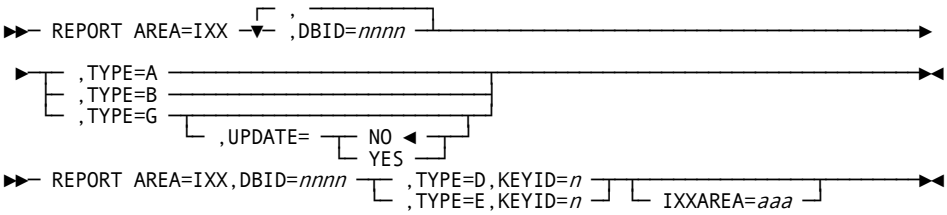
Use the REPORT AREA=IXX option to generate an Index Report. The number of records indexed can be verified by generating an Index Report. You can generate a type A, B, or G Index Area Report at any time. These reports provide information about the entire index whether the database has a single index area or Multi-Dataset Index areas.

The statistics reported in a type G report are used by the SQL Mode for join optimization. If you are doing many INSERTs or DELETEs and you use joins, you want to keep these statistics current. The statistics are updated when the data area is loaded, when the Index Area is rebuilt, or when you execute a type G report with the UPDATE= keyword coded YES.

Generate types D and E at the direction of Technical Support only.

How to Use

You can execute this command in either Single User or Multi-User mode. To generate an Index Area (IXX) Report, execute the REPORT function with the following command format:



Command

REPORT

Invokes the function to produce a report.

Required Keywords

AREA=IXX

Requests the Index Area Report.

DBID=

Identifies the database to be reported.

For Types A, B, and G, DBID= may repeat, in which case each occurrence may specify one or more valid database IDs or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250).

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database in a range not defined is ignored.

Valid Entries:

For Types A, B, and G, one or more valid database IDs or a range of DBIDs as already described.

For Types D and E, the DATACOM-ID of the database

Default Value:

(No default)

,TYPE=

Specifies the type of Index Report to be generated.

A

Specifies a sequence by table name.

B

Specifies a sequence by key ID.

D

Specifies a full dump of the Index Area (IXX).

E

Specifies a dump of the high-level portion of the Index Area (IXX).

G

Specifies a report of the number of records with unique key values.

Valid Entries:

A, B, D, E, G

Default Value:

(No default)

,KEYID=*n*

For TYPE=D,KEYID=*n,* the *n* identifies the specific key with which the dump of the Index Area is to be associated.

For TYPE=E,KEYID=*n,* the *n* identifies the specific key with which the dump of the high-level portion of the Index Area is to be associated.

Note: Use TYPE=D and TYPE=E only at the request of Technical Support.

Valid Entries:

DATACOM-ID of the key

Default Value:

(No default)

Optional Keywords

,UPDATE=

For type G, requests that the entire Index be read to generate current statistics. The report executes in the Multi-User Facility and can occur while any other other Multi-User Facility activity is occurring to this database, including update activity.

Valid Entries:

NO or YES

Default Value:

NO

IXXAREA=*aaa*

IXXAREA= is for types D and E only. For a database that has only the IXX Index Area, the keyword is optional, but if set it must specify IXXAREA=IXX. For a database that has Multi-Dataset Index areas, the keyword is required and must provide the name of one of the index areas (IXX or I*nn*, where *nn* can be 00 to 99).

Valid Entries:

IXX or I00 to I99

Default Value:

(No default)

Type A - Sequenced by Table Name

To generate an Index Area report in alphabetical order by table name, request an IXX report and specify TYPE=A.

Example JCL

The following shows the command to print key related information for database 1, sequenced by table name.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
         REPORT   AREA=IXX,DBID=1,TYPE=A
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT IXX - Type A

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                 Copyright © 2009 CA. All rights reserved.                    *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT  AREA=IXX,DBID=1,TYPE=A

                    FUNCTION=REPORT
       ❷               AREA=IXX
                       DBID=1
                       TYPE=A


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

The spacing in the example is adjusted for display purposes.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX

                    INDEX  POPULATION  REPORT  -  BY TABLE


                      ❹        ❺          ❻           ❼           ❽

                    TABLE    KEY ID     KEY NAME       RECORDS      COMMENTS


                    PAY        001       EMPNO            200

                    PMF        001       EMPNO            200
                               002       STZIP            200

                    PNC        016       PNDAT              5

                    PNM        015       PN                 3

                    POH        011       POLI               5
                               012       VENDR              5
                               013       POHTV              5
                               014       LATE               2

                    POL        011       POLI               4
                               015       PN                 4
```

❹

TABLE - The DATACOM-NAME of the table.

❺

KEY ID - The DATACOM-ID of the key.

❻

KEY NAME - The DATACOM-NAME of the key.

❼

RECORDS - Number of records indexed by this key. If a system failure has occurred during maintenance processing, this number can be inaccurate.

❽

COMMENTS - If the Index for this table is not loaded, a comment is displayed. A comment also appears if the key ID is duplicated within one table. If this is true, the total Index entries for the key ID are reported with the first key name with the same ID. The second (and all others to if any exist) reflect a zero count.

Type B - Sequenced by Key ID

To generate an Index Area report in ascending numeric sequence by Key ID, request an IXX report and specify TYPE=B.

Example JCL

The following shows the command to print key related information for database 1, sequenced by Key ID.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
         REPORT   AREA=IXX,DBID=1,TYPE=B
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT IXX - Type B

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                 Copyright © 2009 CA. All rights reserved.                    *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT  AREA=IXX,DBID=1,TYPE=B

                    FUNCTION=REPORT
       ❷               AREA=IXX
                       DBID=1
                       TYPE=B


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                 Copyright © 2009 CA. All rights reserved.                    *               SPnn
                    ********************************************************************************

                    INDEX  POPULATION  REPORT  -  BY  KEY  ID


                       ❹        ❺         ❻            ❼

                    KEY ID     TABLE    KEY NAME       RECORDS

                      001      PAY       EMPNO             200
                               PMF       EMPNO             200
                                                           400 TOTAL

                      002      PMF       STZIP             200

                      011      POH       POLI                5
                               POL       POLI                4
                                                             9 TOTAL

                      012      POH       VENDR               5

                      013      POH       POHTV               5

                      014      POH       LATE                2

                      015      PNM       PN                  3
                               POL       PN                  4
                                                             7 TOTAL

                      016      PNC       PNDAT               5
```

❹

KEY ID - The DATACOM-ID of the key.

❺

TABLE - The DATACOM-NAME of the table containing the key.

❻

KEY NAME - The DATACOM-NAME of the key.

❼

RECORDS - Number of records indexed by this key. If a system failure has occurred during maintenance processing, this number can be inaccurate.

Types D and E - Dump of the Index Area

When CA Datacom/DB Support requests:

- A dump of the Index Area associated with a specific key, request an IXX report and specify TYPE=D and supply a key ID.
- A dump of the high-level portion of the Index Area associated with a specific key, request an IXX report and specify TYPE=E and supply a key ID.

The following examples are of Type D.

Example JCL

The following shows the command to obtain a dump of the Index Area (IXX) for DBID=1, TYPE=D, and KEYID=1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        REPORT   AREA=IXX,DBID=1,TYPE=D,KEYID=1
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT IXX - Type D

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    ........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT  AREA=IXX,DBID=1,KEYID=1,TYPE=D

                    FUNCTION=REPORT
       ❷               AREA=IXX
                      DBID=1
                      KEYID=1
                      TYPE=D

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

The following sample page of the report is normally followed by several more pages.

```
                                                                DATE:             TIME: 16.31.44    PAGE:


0FE0                                                                    00000001     *                                *
0000  C1C4D961 C4C1E3C1 C3D6D461 C4C261C9   E7E74040 40404040 40404040 40404040     *    CA/DATACOM/DB/IXX           *
0020  00-SAME-
0040  00000000 750C0B01 0F140001 C4D8C3C9   C3E2F140 0002CF99 87050911 10230000     *            DQCICS1             *
0060  00000000 000000A5 0000000F 00000000   00000000 00000000 00000000 00000000     *                                *
0080  00-SAME-
00A0  00000000 C4E2F7D3 D6C1C440 00026E5C   86112019 1327C900 00000000 00000000     *    DS7LOAD    *      I         *
00C0  00-SAME-
0120  00010001 00000000 00010002 00000000   00020002 00000000 00100006 00000000     *                                *
0140  000F0005 00000000 000B0003 00000000   000E0003 00000000 000D0003 00000000     *                                *
0160  000C0003 00000000 000B0004 00000000   000F0004 00000000 00000000 00000000     *                                *
0180  00-SAME-


0FE0                                                                    00000002     *                                *
0000  FF010000 00000000 00000000 0E9D0000   00000000 00070000 00030001 F0F0F1F1     *                           0011*
0020  F1020000 00000401 01000000 05020101   00000006 0B010100 0000070C 01010000     *1                               *
0040  00080D01 01000000 090E0101 0000000A   0F010100 00000B10 00020000 000C038F     *                                *
0060  00020000 000D999A 01010000 000E9B00   02000000 00FFFF00 00000000 00000000     *                                *
0080  00-SAME-


0FE0                                                                    00000003     *                                *
0000  00010100 00000004 00000000 06320000   00010027 00DE0000 00006F00 05000000     *                                *
0020  0001F0F0 F1F1F100 18000501 F0F0F0F0   F1000100 00000201 00020000 00020100     *  00111     00001               *
0040  14040102 F2000100 00000202 00020000   00020200 14040103 F3000100 00000203     *    2                   3       *
0060  00020000 00020300 14040104 F4000100   00000204 00020000 00020400 14040105     *            4                   *
0080  F5000100 00000205 00020000 00020500   14040106 F6000100 00000206 00020000     *5                   6           *
00A0  00020600 14040107 F7000100 00000207   00020000 00020700 14040108 F800010q     *        7                   8   *
00C0  00000208 00020000 00020800 14040109   F9000100 00000209 00020000 00020900     *                9               *
00E0  1503020A F1F00001 00000002 0A000200   0000020A 00140401 0BF10001 00000002     *    10                  1       *
0100  0B000200 0000020B 00140401 0CF20001   00000002 0C000200 0000020C 00140401     *             2                  *
0120  0DF30001 00000002 0D000200 0000020D   00140401 0EF40001 00000002 0E000200     * 3                   4          *
0140  0000020E 00140401 0FF50001 00000002   0F000200 0000020F 00140401 10F60001     *         5                    6  *
0160  00000002 10000200 00000301 00140401   11F70001 00000002 11000200 00000302     *                  7             *
0180  00140401 12F80001 00000002 12000200   00000303 00140401 13F90001 00000002     *     8                    9      *
01A0  13000200 00000304 00150302 14F2F000   01000000 03010002 00000003 05001404     *              20             *
01C0  0115F100 01000000 03020002 00000003   06001404 0116F200 01000000 03030002     *  1                   2        *
01E0  00000003 07001404 0117F300 01000000   03040002 00000003 08001404 0118F400     *          3                   4 *
0200  01000000 03050002 00000003 09001404   0119F500 01000000 03060002 00000003     *                  5             *
0220  0A001404 011AF600 01000000 03070002   00000003 0B001404 011BF700 01000000     *      6                   7     *
0240  03080002 00000003 0C001404 011CF800   01000000 03090002 00000003 0D001404     *             8                  *
0260  011DF900 01000000 030A0002 00000003   0E001503 021EF3F0 00010000 00030B00     *  9                   30       *
0280  02000000 04010014 04011FF1 00010000   00030C00 02000000 04020034 040120F2     *           1                   2*
02A0  00010000 00030D00 02000000 04030014   040121F3 00010000 00030E00 02000000     *                  3             *
02C0  04040014 040122F4 00010000 00030F00   02000000 04050014 040123F5 00010000     *       4                   5    *
02E0  00031000 02000000 04060014 040124F6   00010000 00031100 02000000 04070014     *              6               *
0300  040125F7 00010000 00031200 02000000   04080014 040126F8 00010000 00031300     *   7                   8       *
0320  02000000 04090014 040127F9 00010000   00040100 02000000 040A0015 030228F4     *          9                   4*
```

Type G - Number of Records With Unique Key Values

To generate an Index Area report with statistics on records with unique key values, request an IXX report and specify TYPE=G.

The statistics reported in a type G report are used by the SQL Mode for join optimization. If you are doing many INSERTs or DELETEs and you use joins, you want to keep these statistics current. The statistics are updated when the data area is loaded, when the Index Area is rebuilt, or when you execute a type G report with the UPDATE= keyword coded YES.

Example JCL

The following shows the command to print key related information for database 1, sequenced by table name.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.index',,DA                   Index Area for DBID 001
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        REPORT   AREA=IXX,DBID=1,TYPE=G,UPDATE=YES
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT IXX - Type G

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT  AREA=IXX,DBID=1,TYPE=G,UPDATE=YES

                    FUNCTION=REPORT
       ❷               AREA=IXX
                       DBID=1
                       UPDATE=YES
                       TYPE=G

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *     Release: 12
                    *                 Copyright © 2009 CA. All rights reserved.                    *              SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX

                    INDEX  POPULATION  REPORT  -  BY TABLE

                      ❹       ❺         ❻            ❼            ❽            ❾             ❿

                                                     TOTAL        UNIQUE
                    TABLE    KEY ID     KEY NAME    RECORDS      RECORDS      DATE/TIME       COMMENTS

                    PAY        001      EMPNO          200                5/25/2003,09.15.44
                                                                      200
                    PMF        001      EMPNO          200                5/25/2003,09.15.48
                                                                      200
                               002      STZIP          200                5/25/2003,09.15.48
                                                                        6
                                                                        8
                    PNC        016      PNDAT            5                5/25/2003,09.15.51
                                                                        3
                                                                        5
                    PNM        015      PN               3                5/25/2003,09.15.51
                                                                        3
                    POH        011      POLI             5                5/25/2003,09.15.51
                                                                        5
                                                                        5
                               012      VENDR            5                5/25/2003,09.15.51
                                                                        3
                               013      POHTV            5                5/25/2003,09.15.51
                                                                        4
                               014      LATE             2                5/25/2003,09.15.51
                                                                        1

                    POL        011      POLI             4                5/25/2003,09.15.51
                                                                        3
                                                                        4
                               015      PN               4                5/25/2003,09.15.51
                                                                        3
```

❹

TABLE - The DATACOM-NAME of the table.

❺

KEY ID - The DATACOM-ID of the key.

❻

KEY NAME - The DATACOM-NAME of the key.

❼

TOTAL RECORDS - Number of records indexed by this key at LOAD time, RETIX time, or currently, if UPDATE=YES is specified.

❽

UNIQUE RECORDS - Number of records with distinct values for this key field or key field combination. The first value given is the number of records with distinct values for the high-order key field. The next value given is the number of records with distinct values for the high-order key field and the adjacent key field and so on until the last value, which gives the number of records with distinct values for the entire key (all fields).

For example, if you have a key named NAMEADDR made up of fields: LNAME, MI, FNAME, and ADDR, the key statistics would be reported as follows:

```
Value Reported    Records With Distinct Values for:
                  &bxul.&bxh.&bxh.&bxh.&bxh.&bxh.&bxur.
  1st value       &bxv.LNAME&bxv.MI FNAME ADDR
                  &bxll.&bxh.&bxh.&bxh.&bxh.&bxh.&bxlr.

                  &bxul.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxur.
  2nd value       &bxv.LNAME MI&bxv.FNAME ADDR
                  &bxll.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxlr.

                  &bxul.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxur.
  3rd value       &bxv.LNAME MI FNAME&bxv.ADDR
                  &bxll.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxlr.

                  &bxul.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxur.
  4th value       &bxv.LNAME MI FNAME ADDR&bxv.
                  &bxll.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxh.&bxlr.
```

❾

DATE/TIME - Date and time of the LOAD or RETIX when this data was compiled, or the date and time that this report was executed, if UPDATE=YES is specified.

❿

COMMENTS - If the Index for this table is not loaded, a comment is displayed.

If the key ID is duplicated within one table, a comment is displayed. In this case, the total Index entries for the key ID are reported with the first key name with that ID. The second (and any others) show a zero count.

Log Area (LXX) Report

The Log Area Report prints information about the Log Area, such as:

- Whether the Log File is open
- Whether the Log File is spilling or is in a spill pending status
- Whether the Log File has any errors
- Whether the Multi-User startup option LOGRCV is selected
- The highest Transaction Sequence Number (TSN)
- The cycle start date and time
- Information about the blocks relative to the cycle
- The log pending parameter

When to Use

Use the REPORT AREA=LXX option to obtain information about the Log Area. Specifically, you can use this report to determine the highest Transaction Sequence Number (TSN).

You can request a Log Area report any time. The information obtained reflects information stored on disk, not necessarily the current information the Multi-User Facility is using.

How to Use

You can execute this command in either Single User or Multi-User mode. To obtain a Log Area Report, execute the REPORT function with the following command format.

Command

REPORT

Invokes the function to produce a report.

Required Keywords

AREA=LXX

Requests the Log Area Report.

Example JCL

The following shows the command to generate a Log Area report.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL LXX,'lxx',,DA                              Log Area data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
          REPORT   AREA=LXX
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT LXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT AREA=LXX

                    FUNCTION=REPORT
       ❷                 AREA=LXX



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing are presented in this area.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                             Multi-User Facility                              *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                                          LOG FILE (LXX) OUTPUT
               ❹
                    LOG FILE OPEN -  NO
                    LOG FILE SPILLING -  NO
                    LOG FILE SPILL PENDING -  NO
                    MASTER LIST RCV OPTION - YES
                    LOG FILE HIGHEST TRANSACTION SEQUENCE NUMBER - DBDC2C2A
                    CYCLE START DATE -   5/06/2009
                    CYCLE START TIME -   7.14.33
                    FIRST BLOCK USED THIS CYCLE - 0001C800
                    CURRENT LOG BLOCK - 0001C8FF
                    PHYSICAL BLOCKS USED THIS CYCLE -            0
                    RESTART FORWARD SCAN BLOCK - 0001C800
                    LOG PENDING PARAMETER -     100
                    LOG FILE BLOCK SIZE -       32,760
                    PHYSICAL BLOCK SIZE -  32,760
                    LOG VARIABLE - YES
                    PHYSICAL BLOCKS PER LOGICAL BLOCK -    1
```

❹

This section of the report provides messages that indicate the status of the Log Area.

LOG FILE HIGHEST TRANSACTION SEQUENCE NUMBER indicates the last transaction sequence number in the Log Area.

VARIABLE indicates if the LXX is initialized as variable (YES) or not (NO), specified by the DBUTLTY INIT AREA=LXX VARIABLE= option.

The date and time (*ccyymmddhhmmss*) of the LXX INIT, FXX INIT, and LAST LOG RECORD indicates when each was last initialized, as follows:

- *cc* : century
- *yy* : year
- *mm:* month
- *dd:* day
- *hh:* hour
- *mm:* minute
- *ss:* second

The date and time of the INITs is that which DBUTLTY was using when executed. Prior to Version 12.0, this date and time was adjusted relating to current daylight saving time. It is printed as binary zeros if the LXX or FXX was not subject to INIT in Version 11.0 (also known as r11) SP2 or later. Following the date and time is the store clock value (in hex, *cccccccccccccc*, for example C23D3418E97E5B82) at the time of the INIT.

LXX RESET - NO indicates that the LXX in this example has not been reset since it was last initialized. If the LXX had been RESET, the NO would be replaced with the date and time (*ccyymmddhhmmss*) of the RESET, followed by the store clock value (*cccccccccccccc*) at the time of the RESET.

The FXX TOTAL TASKS line shows the Force Area total tasks that are allowed to execute. This is the number that restricts whether MUF can enable the line (below the line) by specifying the number of total tasks that are allowed, the most that have been used since the last INIT, and the current number of system tasks. The number of user tasks allowed are reduced by the number of system tasks.

The FXX USER TASKS line is the number of user tasks that are allowed, based upon the current total tasks.

The next line gives information about the meaning of user tasks. When a user does an INIT of the FXX with TASKS=, this number is the user tasks, to which CA Datacom/DB adds the current number of system tasks to arrive at a number of total tasks. Because, as already mentioned, the number of user tasks allowed are reduced by the number of system tasks, if the number of system tasks changes, the number of user tasks can decrease.

The LAST LOG RECORD line provides the date and time (*ccyymmddhhmmss*) and store clock value (*ccccccccccccccc*) of the last record logged in the prior MUF. It is zeros for the first MUF after an INIT. This line is provided for information only. If the MUF did a full restart, the scan process looked for a step-down condition to stop. If a step-down is found, the date and time and store clock value of the step-down is provided, preceded by an * as shown in the sample report. If the step-down occurred not to a valid log record but to a binary zero block (it is zeros for the first MUF after an INIT), the message would instead be STEP DOWN - BINARY ZERO BLOCK. If the step-down occurred to an invalid record, the message would instead be STEP DOWN - INCOMPLETE RECORD (this condition could occur if the block data was being sent down the channel when a sever machine condition interrupted the flow and what was written to DASD was incomplete, which might or might not be possible with current hardware).

REFGROUP Report

A DBUTLTY REFGROUP report allows you to determine the current data organization in a database by accessing each data row within a specified table by reference groupings. You use a reference group parameter (REFGROUP=) to specify the number of data blocks that are to be considered as a group. For information, see REFGROUP= in the list of Required Keywords. REFGROUPs are used to analyze (report) and reorganize (using the OLREORG function for online database reorganization) data rows on a reference group by reference group basis. Access for the REFGROUP report option is through the Multi-User Facility. For information on the OLREORG function, see OLREORG (Reorganize a Database Online).

You can select two types of REFGROUP reports, summary reports (TYPE=S) and detail reports (TYPE=D or TYPE=K). For information on specifying TYPE=, see TYPE= in the list of Required Keywords.

Restrictions

The hardware environment in which the REFGROUP report is run must have relative branching support.

The area in which the table (that you specify for the REFGROUP report) resides must be loaded with URI=YES.

You must have logging turned on in order to generate a REFGROUP report.

Security Considerations

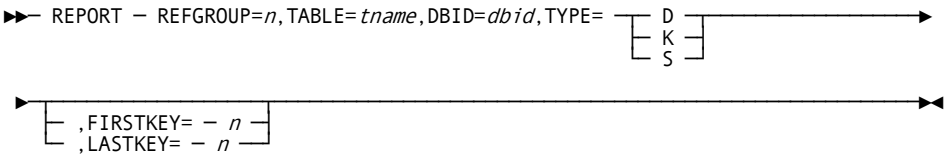
The DBUTLTY external resource name for the REFGROUP report is REPORT.REFGROUP. At the table level it uses the read authorization.

When to Use

Use the REFGROUP report when you want to determine the current data organization by reference group in order to decide whether to execute the online database reorganization function (OLREORG), described beginning on page OLREORG (Reorganize a Database Online))

How to Use

To obtain a REFGROUP report, execute the REPORT function with the following command format:



Command

REPORT

Invokes the REPORT function.

Required Keywords

REFGROUP=*n*

Specify the number of blocks to treat as a reference group.

Valid Entries:

2 through 100

Default Value:

(No default)

,TABLE=*tname*

Specify the name of the table that you want to use. The area in which the table resides must be loaded with URI=YES. If you attempt to generate a REFGROUP report and the area was not loaded with URI, an error occurs and a return code is generated.

Valid Entries:

a valid table name

Default Value:

(No default)

,DBID=

Specify the name of the DBID that you want to use.

Valid Entries:

a valid DBID

Default Value:

(No default)

,TYPE=

Three types of reports are possible, indicated with D, K, or S. Summary reports a (TYPE=S) echo the DBUTLTY parameter specifications (standard within DBUTLTY) and process the entire data table to produce several summary rows. Detail reports (TYPE=D) echo the DBUTLTY parameter specifications (standard within DBUTLTY), process the entire data table, and create one detail row for each reference group as well as several summary rows. TYPE=K detail reports also include key value information.

Valid Entries:

D, K, or S

Default Value:

(No default)

Optional Keywords

,FIRSTKEY= and/or ,LASTKEY=

You can use FIRSTKEY= and/or LASTKEY= to limit REFGROUP to a segment of the specified key. We recommend you allow FIRSTKEY=/LASTKEY= to default. REFGROUP normally defaults to include the full specified key range from low values to high values. FIRSTKEY= overrides the default starting position while LASTKEY= overrides the default ending position, limiting what is processed by REFGROUP. If one keyword is specified and the other is not specified, a default value is selected for the missing keyword. Normal CA Datacom key lengths are from 1 through 180 bytes. The key value data entered for FIRSTKEY= and LASTKEY= can be from 0 (zero) through 59 bytes long. You only need to code the number of positions of the key value that are significant to you, but be aware that the value coded is *left justified regardless of key field data type.* The utility pads the low order or remaining positions of the key value with low values on FIRSTKEY= and high values on LASTKEY=. To use the entire key in one step, specify neither FIRSTKEY= nor LASTKEY=. Alternately, you could specify FIRSTKEY=00 and/or LASTKEY=00 to cause processing to start at low values and end at high values. The length of zero indicates that the padded key values (low values and high values) are to be used.

The data is not edited or interpreted. Therefore, if the key value data you need to enter is in binary, you must set up the JCL using a hexadecimal mode display and enter the key value data in hexadecimal. This data can contain blanks, commas, and any other special characters. CA Datacom/DB cannot edit this data for syntax, so the control statement can *look* invalid, especially when it includes blanks or commas or key value data containing English words. For example, FIRSTKEY=12121212121212 means the length is 12, key value is 121212121212. Another example, FIRSTKEY=10FIRSTKEY10 means the length is 10, key value is FIRSTKEY10.

Valid Entries:

A 2-byte length (in a range of 00 through 59) followed by a character string of key value bytes (for the length specified) that represents the key value.

Default Value:

The starting/ending position of the full key range from low to high values.

Example JCL (Summary REFGROUP Report TYPE=S)

The following shows the command to obtain a REFGROUP report of TYPE=S.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
*$$ JOB ...See the previous note and JCL Requirements (see page 34).

*$$ LST ...

//JOB name
//EXEC DBUTLTY,SIZE=750K
        REPORT REFGROUP=2,DBID=2,TABLE=ALS,TYPE=S
/*
/&
*$$ EOJ
```

Sample Report: REFGROUP Summary

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                     CONTROL CARD(S)

 .........1.........2.........3.........4.........5.........6.........7.........8
        ❶             REPORT   REFGROUP=2,DBID=2,TABLE=ALS,TYPE=S
                     FUNCTION=REPORT
        ❷               DBID=00002
                        REFGROUP=00002
                        TABLE=ALS
                        TYPE=S
        ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
Base:     2         ******************************************************************************** Directory: PRODCXX

                    Table Block Usage Report by Reference Group
   ❶      ❷     ❸           ❹      ❺      ❻                                      ❼
            Rows   Ref       1     2     3     4     5     6     7     8     9   10+   15+   20+   25+   30+   40+   50+
Groups  Rows /Blk  Blks     Row  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows
698181                  609,196      71,784      27,918      17,238      11,804      31,738      27,235       7,293
  12,499,845    8         224,261      41,954      21,943      13,965      43,569      27,540     218,924

    ❽
```

❶

Groups (698181 in this example) is the total number of REFGROUPs for this table, based on the value coded for REFGROUP.

❷

Rows (12,499,845 in this example) is the total rows in the table.

❸

Rows/Blk (8 in this example) is the average rows within a REFGROUP per block.

❹

1 Row (609,196 in this example) is the total number of data blocks with only 1 row for a REFGROUP.

❺

2 Rows (224,261 in this example) is the total number of data blocks with 2 rows for a REFGROUP.

❻

3 Rows (71,764 in this example) is the total number of data blocks with 3 rows for a REFGROUP, and so on through through 9 Rows (11,804 in this example), the total number of data blocks with 9 rows for a REFGROUP.

❼

The remainder of the Rows descriptions are as follows:

10+ Rows

(43,569 in this example) is the total number of data blocks with 10 through 14 rows for a REFGROUP.

15+ Rows

(31,738 in this example) is the total number of data blocks with 15 through 19 rows for a REFGROUP, and so on through 25+ Rows (27,235 in this example), the total number of data blocks with 25 through 29 rows for a REFGROUP.

30+ Rows

(218,924 in this example) is the total number of data block with 30 through 39 rows for a REFGROUP.

40+ Rows

(7,293 in this example) is the total number of data block with 40 through 49 rows for a REFGROUP.

50+ Rows

(blank in this example) is the total number of data block with 50 or more rows for a REFGROUP.

❽

If a table is not defined with compression there is also a line below those shown in the example report. That line, "Maximum Possible Rows per Block," includes a number that can be used to help analyze the values printed for rows and to know how well ordered your data is by REFGROUP.

Example JCL (Detail REFGROUP Report TYPE=D)

The following shows the command to obtain a REFGROUP report of TYPE=D.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
*$$ JOB ...See the previous note and JCL Requirements (see page 34).

*$$ LIST ...

//JOB name
//EXEC DBUTLTY,SIZE=750K
        REPORT REFGROUP=2,DBID=2,TABLE=FLD,TYPE=D
/*
/&
*$$ EOJ
```

Sample Report: REFGROUP Detail

This report is similar to the type S report except there is one detail line per REFGROUP. This allows you to see if only certain parts of your data are not well organized by native key within a REFGROUP.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                    CONTROL CARD(S)
 .........1.........2.........3.........4.........5.........6.........7.........8
       ❶              REPORT   REFGROUP=2,DBID=2,TABLE=FLD,TYPE=D
                     FUNCTION=REPORT
       ❷                DBID=00002
                       REFGROUP=00002
                       TABLE=FLD
                       TYPE=D
       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
Base:     2         ******************************************************************************** Directory: PRODCXX

                    Table Block Usage Report by Reference Group
   ❹   ❺    ❻     ❼      ❽
  Hex       Rows   Ref      1     2     3     4     5     6     7     8     9   10+   15+   20+   25+   30+   40+   50+
Group  Rows /Blk  Blks    Row  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows
    1    55   27     2      .     .     .     .     .     .     .     .     .     1     .     .     .     .     1
    2    27   13     2      1     .     .     .     .     .     .     .     .     .     .     1
    3    20   10     2      .     1     .     .     .     .     .     .     .     .     1
    4    20   10     2      1     .     .     .     .     .     .     .     .     .     1
    5     6    3     2      .     1     .     1
    6    23   11     2      1     .     .     .     .     .     .     .     .     .     .     1
    7     2    1     2      2
    8    41   20     2      .     .     .     .     .     .     .     .     .     1     .     .     .     1
    9    11    5     2      1     .     .     .     .     .     .     .     .     .     1
    A    79   39     2      .     .     .     .     .     .     .     .     .     .     .     .     .     1     1
    B    42   21     2      .     1     .     .     .     .     .     .     .     .     .     .     .     .     1
    C    81   40     2      .     .     .     .     .     .     .     .     .     .     .     .     .     .     2
    D    15    7     2      1     .     .     .     .     .     .     .     .     1
    E     2    1     2      2
    F    59   29     2      .     .     .     .     .     .     .     .     .     .     .     1     .     1
   10    77   38     2      .     .     .     .     .     .     .     .     .     .     .     .     .     2
   11    78   39     2      .     .     .     .     .     .     .     .     .     .     .     .     .     2
    .
    .
    .
AA743    36   18     2      .     1     .     .     .     .     .     .     .     .     .     .     .     .     1
AA744    38   19     2      .     1     .     .     .     .     .     .     .     .     .     .     .     .     1
AA745    35   17     2      1     .     .     .     .     .     .     .     .     .     .     .     .     .     1
 ❾
Groups---------------------------------------------Totals ----------------------------------------------------------
698181              609,196      71,784      27,918      17,238      11,804      31,738      27,235       7,293
 12,499,845    8            224,261      41,954      21,943      13,965      43,569      27,540     218,924

 ❿
```

❹

Hex Group is an ascending counter starting with 1 assigned to each REFRGOUP. It is a hexadecimal number.

❺

Rows is the total number of rows in this REFGROUP.

❻

Rows/Blk is the average rows per block for this REFGROUP.

❼

Ref Blks shows the blocks in this REFGROUP. This is the same as the value coded for REFGROUP, except for the last REFGROUP.

❽

1 Row through 50+ Rows has the same explanation as was provided in the TYPE=S REFGROUP report, except that the TYPE=D REFGROUP report is reported for each REFGROUP.

❾

At the end of the TYPE=D REFGROUP report are the totals that summarize the report.

❿

If a table is not defined with compression there is also a line below those shown in the example report. That line, "Maximum Possible Rows per Block," includes a number that can be used to help analyze the values printed for rows and to know how well ordered your data is by REFGROUP.

Example JCL (Detail REFGROUP Report TYPE=K)

The following shows the command to obtain a REFGROUP report of TYPE=K. A TYPE=K detail report includes key value information.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
*$$ JOB ...See the previous note and JCL Requirements (see page 34).

*$$ LIST ...

//JOB name

//DLBL CXX.'cxx' CXX dataset
//EXEC DBUTLTY,SIZE=75_K
        REPORT REFGROUP=2,DBID=2,TABLE=FLD,TYPE=K
/*
/&
*$$ EOJ
```

Sample Report: REFGROUP Detail with Key Value Information

This report is similar to the type D report except that following the detail lines there is a first key value printed for each REFGROUP.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                    CONTROL CARD(S)
 .........1.........2.........3.........4.........5.........6.........7.........8
       ❶              REPORT   REFGROUP=2,DBID=2,TABLE=FLD,TYPE=K
                     FUNCTION=REPORT
       ❷                DBID=00002
                       REFGROUP=00002
                       TABLE=FLD
                       TYPE=K
       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
Base:     2         ******************************************************************************** Directory: PRODCXX

                    Table Block Usage Report by Reference Group
   ❹   ❺    ❻     ❼      ❽
  Hex        Rows   Ref      1     2     3     4     5     6     7     8     9   10+   15+   20+   25+   30+   40+   50+
Group  Rows /Blk  Blks    Row  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows  Rows
    1    55   27     2      .     .     .     .     .     .     .     .     .     1     .     .     .     .     1
    2    27   13     2      1     .     .     .     .     .     .     .     .     .     .     1
    3    20   10     2      .     1     .     .     .     .     .     .     .     .     1
    4    20   10     2      1     .     .     .     .     .     .     .     .     .     1
    5     6    3     2      .     1     .     1
    6    23   11     2      1     .     .     .     .     .     .     .     .     .     .     1
    7     2    1     2      2
    8    41   20     2      .     .     .     .     .     .     .     .     .     1     .     .     .     1
    9    11    5     2      1     .     .     .     .     .     .     .     .     .     1
    A    79   39     2      .     .     .     .     .     .     .     .     .     .     .     .     .     1     1
    B    42   21     2      .     1     .     .     .     .     .     .     .     .     .     .     .     .     1
    C    81   40     2      .     .     .     .     .     .     .     .     .     .     .     .     .     .     2
    D    15    7     2      1     .     .     .     .     .     .     .     .     1
    E     2    1     2      2
    F    59   29     2      .     .     .     .     .     .     .     .     .     .     .     1     .     1
   10    77   38     2      .     .     .     .     .     .     .     .     .     .     .     .     .     2
   11    78   39     2      .     .     .     .     .     .     .     .     .     .     .     .     .     2
    .
    .
    .
AA743    36   18     2      .     1     .     .     .     .     .     .     .     .     .     .     .     .     1
AA744    38   19     2      .     1     .     .     .     .     .     .     .     .     .     .     .     .     1
AA745    35   17     2      1     .     .     .     .     .     .     .     .     .     .     .     .     .     1
  ❾
Groups---------------------------------------------Totals ----------------------------------------------------------
698181               609,196      71,784      27,918      17,238      11,804      31,738      27,235       7,293
 12,499,845    8             224,261      41,954      21,943      13,965      43,569      27,540     218,924

  ❿
```

❹

Hex Group is an ascending counter starting with 1 assigned to each REFRGOUP. It is a hexadecimal number.

❺

Rows is the total number of rows in this REFGROUP.

❻

Rows/Blk is the average rows per block for this REFGROUP.

❼

Ref Blks shows the blocks in this REFGROUP. This is the same as the value coded for REFGROUP, except for the last REFGROUP.

❽

1 Row through 50+ Rows has the same explanation as was provided in the TYPE=S REFGROUP report, except that the TYPE=D REFGROUP report is reported for each REFGROUP.

❾

At the end of the TYPE=D REFGROUP report are the totals that summarize the report.

❿

If a table is not defined with compression there is also a line below those shown in the example report. That line, "Maximum Possible Rows per Block," includes a number that can be used to help analyze the values printed for rows and to know how well ordered your data is by REFGROUP.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                   Copyright © 2009 CA. All rights reserved.                  *               SPnn
Base:     2         ********************************************************************************* Directory: PRODCXX
                    Key Values by Reference Group
 Group  First Key Value

     1  FLDACCTS
        CDCCCCEE4444444444444444444444444444000044444444444444444444
        6341333200000000000000000000000000001000000000000000000000000
     2  FLDAGGREGATE                          AGR-REQRD
        CDCCCCDCCCEC4444444444444444444444444000FCCD6DCDDC44444444444
        6341779571350000000000000000000000000062C17909589400000000000
     3  FLDDATABASE                           DATABASE-USER
        CDCCCECCCEC4444444444444444444444444009CCECCCEC6EECD4444444
        6344131212500000000000000000000000000C13413121250425900000000
     4  FLDDDMT3002
        CDCCCDEFFFF4444444444444444444444444000044444444444444444444
        6344443300200000000000000000000000000010A00000000000000000000
     5  FLDDDMT5100
        CDCCCDEFFFF4444444444444444444444444000044444444444444444444
        6344443510000000000000000000000000000300000000000000000000000
     6  FLDDIR_DATASET
        CDCCCD6CCECECE4444444444444444444444000544444444444444444444
        634499D413125300000000000000000000000103000000000000000000000
     7  FLDFIELD
        CDCCCCDC4444444444444444444444444444010D44444444444444444444
        6346953400000000000000000000000000011C00000000000000000000
              .
              .
              .
```

This last page of a TYPE=K report is where it differs from a TYPE=D REFGROUP report, in that following the detail lines there is a first key value printed for each REFGROUP. The hexadecimal group number is printed so that you can correlate this with the statistics for a particular REFGROUP. This can be useful if you later choose to run an OLREORG report against only part of the table using either FIRSTKEY or LASTKEY or both.

The actual key values print in character form. Optionally, if there is non-character data, two lines with the hexadecimal values for this key are printed. The key value printed is limited to 59 bytes and is then truncated. This matches the limitations on FIRSTKEY and LASTKEY.

Statistics and Diagnostics Area (PXX) Report

The Statistics and Diagnostics Area (PXX) Report prints the data contained in the PXX. Statistics, dumps, and other information are stored in the PXX in a unique format.

Note: You can obtain information about status, performance, and definitional data in real time through SQL access to special system tables. These dynamic tables allow easy access of the data from any platform while the MUF is running. See the *CA Datacom/DB System Tables Reference Guide* for information.

When to Use

Use the REPORT AREA=PXX option to generate a PXX Report. Use the PXX Report function either to produce formatted dumps for use as debugging tools or to generate reports of the statistics data. You can request a Statistics and Diagnostics Area report at any time.

Information Available for Reporting

All information physically in the PXX is available to print since the MUF was last enabled or the last time the PXX was cleared. If the PXX becomes full, no new information is added until after it is cleared. Job and system statistic information in the report is available based upon the PXXSTATS Multi-User startup options. If PXXSTATS is set to DETAIL (or allowed to default), the MUF writes the information at every URT close where the PXX was not currently full. If PXXSTATS is set to EOJPRT, the MUF writes the summary statistics to the MUF EOJ report instead of to the PXX. If PXXSTATS is set to EOJ, the MUF writes one set of statistics to the PXX during the EOJ of the MUF where it is available until the next time the MUF is enabled.

Specifying PXXSTATS EOJ results in the MUF statistics record being written only one time and only at EOJ. This means that if the MUF is abended or canceled and there is no EOJ, those statistics are not available to the DBUTLTY function REPORT AREA=PXX. Attempts to use the REPORT function to print those statistics therefore generates DATABASE NOT ENDED error messages instead of written statistics.

If the MUF ends normally with PXXSTATS EOJ specified, at EOJ a single set of statistical records are written to the PXX. Running the DBUTLTY REPORT AREA=PXX function then gives different results depending on the combinations of options you specify. With PXXSTATS EOJ we recommend using the following:

```
DBUTLTY REPORT AREA=PXX,SYSSTAT=SUMMARY,JOBSTAT=SUMMARY
```

This gives SYSTEM summary STATS (database requests down to requests pending) plus JOBSTAT summaries (AREA and TABLE requests and I/O statistics). Other possible option specifications include the following:

1. DBUTLTY REPORT AREA=PXX,SYSSTAT=SUMMARY gives SYSTEM summary STATS but not JOBSTAT summaries.
2. DBUTLTY REPORT AREA=PXX,SYSSTAT=DETAIL gives one copy of SYSTEM summary STATS with a jobname matching the MUF, plus JOBSTAT summaries for the same jobname, then the SYSTEM summary STATS again, then JOBSTAT summaries again. In this case, statistics are unnecessarily duplicated.

For detailed information about Multi-User startup options, see the *CA Datacom/DB Database and System Administration Guide*.

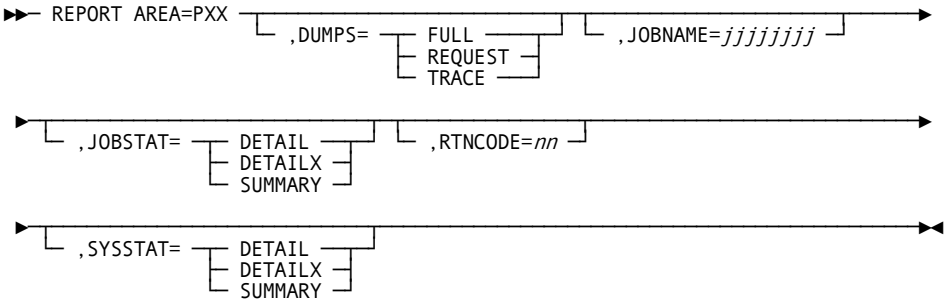
In addition to the statistics and diagnostic information routinely generated by CA Datacom/DB, use the DBUTLTY COMM ALTER function to request that CA Datacom/DB generate trace information, dumps, and messages to assist in problem determination. See COMM ALTER (Modify Error Table) (see page 116) for details on the COMM ALTER function.

Length of Time to Execute

Each Statistics and Diagnostics Area (PXX) report option that you request causes the entire Statistics and Diagnostics Area to be read. It is a keyed file. If your PXX data is allocated in a large area, elapsed time to execute this function can be long.

How to Use

To obtain a Statistics and Diagnostics Area Report, execute the REPORT function with the following command format:



Command

REPORT

Invokes the function to produce a report.

Required Keywords

Note: You must include at least one of the keywords:DUMPS, JOBSTAT, or SYSSTAT.

AREA=PXX

Requests the Statistics and Diagnostics Area Report.

,DUMPS=

Specifies that formatted dumps are to be printed.

FULL

Dumps Master List, buffer areas, Request Area, and Compound Boolean Selection Facility (CBS) Diagnostic Report.

The process of producing dumps has changed. Dumps are produced using new (in Version 12.0) and faster memory management methods that support 64-bit addressing. Because of this enhanced processing and to simplify it, the following changes were made:

Dump Addresses Format Changes

The formatting of dumps has changed in Version 12.0 to show addresses in 64-bit memory that start with one byte, followed by an underscore, then the address within 64-bit memory. This new way of formatting dump addresses appears in both the DBUTLTY REPORT AREA=PXX and in the Master List dump output directed to the PXXML by the SYSOUT Multi-User startup option.

STATBFR Startup Option Removed

You are not, in Version 12.0, allowed to specify STATBFR in the Multi-User startup. The STATBFR Multi-User startup option was previously used to specify the size of the buffer that accumulates system and job statistics, but STATBFR was removed in Version 12.0. Complete statistics are now always provided when requested.

.ECSA Changes

The Master List in the ECSA is now labeled DB_ML_ECSA. If a communication area exits for DB_ML_ECSA, its label is COMMECSA.

Message DB02406I Changed

To support the change in dump processing in Version 12.0, message DB02406I has changed. The message text could previously contain a variable, replaced with a ML or a BFR, that could possibly be separated by another PXX dump. The multiple parts of the process, and the variable in the message for ML or BFR, have been removed in Version 12.0. The process now has a single completion. In addition, text describing the error has been added to the end of the message. For details, see the *CA Datacom/DB Message Reference Guide*.

REQUEST

Prints only the Request Area portion of the dumps.

TRACE

Produces only the Compound Boolean Selection Facility (CBS) Diagnostic Report and the SQL trace information requested through the DBUTLTY COMM ALTER function. See Reporting on the Compound Boolean Selection Facility (see page 594) and COMM ALTER (Modify Error Table) (see page 116).

Valid Entries:

FULL, REQUEST, TRACE

Default Value:

No dumps are printed.

,JOBSTAT=

Indicates that a job statistics report is to be printed.

Note: To get JOBSTAT information, either comment out the PXXSTATS Multi-User startup option or specify it as PXXSTATS DETAIL. Detailed information about Multi-User startup options can be found in the *CA Datacom/DB Database and System Administration Guide*.

DETAIL

Performs a top of form and prints JOBSTAT detail, causing each JOBSTAT to start on a new page.

DETAILX

Skips three lines and prints JOBSTAT detail, causing JOBSTAT information to print on the same page with other information.

SUMMARY

Performs a top of form and prints only the summary information from each job close.

Valid Entries:

DETAIL, DETAILX, or SUMMARY

Default Value:

No JOBSTAT information is printed.

,SYSSTAT=

Indicates that system statistics information is to be printed. The value you code for this keyword specifies the format of the Statistics Report. System statistics are in the Report Area (PXX) for each CLOSE request. These statistics are cumulative and reflect information from the time the system begins running.

Note: The frequency at which MUF statistics records are written (if any), are controlled by URT closings and the PXXSTATS Multi-User startup option. For detailed information about Multi-User startup options, see the *CA Datacom/DB Database and System Administration Guide*.

DETAIL

Performs a top of form and prints system statistics for each CLOSE request.

DETAILX

Skips three lines and prints system statistics for each CLOSE request, causing SYSSTAT information to print on the same page with other information.

SUMMARY

Performs a top of form and prints the system statistics from the last CLOSE record in the Report Area (PXX).

Valid Entries:

DETAIL, DETAILX, SUMMARY

Default Value:

No SYSSTAT information is printed.

Note: When you request a PXX report with JOBSTAT DETAIL that specifies either a JOBNAME= or RTNCD= keyword, be aware that both job and summary statistics are generated.

Optional Keywords

,JOBNAME=

Specifies the name of the job for any dumps and statistics that are to be reported.

Valid Entries:

Any valid job name

Default Value:

Statistics and dumps are printed for all jobs.

,RTNCODE=

Specifies that dumps are to be printed only if the named return code is found in the Request Area. The RTNCODE= parameter is active when the DUMPS=FULL or DUMPS=REQUEST keyword. If DUMPS=TRACE is specified, RTNCODE= is ignored.

Valid Entries:

Any CA Datacom/DB return code (2-character numeric)

Default Value:

(No default)

Generating a PXX Report

To print a report showing job and system statistics contained in the Statistics and Diagnostics Area, request a PXX report with values specified for JOBSTAT= and SYSSTAT=.

Example JCL

The following shows the command to print detail job and summary system statistics.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
* $$ JOB ...           See the previous note and :ref refid=jclreq..
* $$ LST ...
// JOB name
// ASSGN SYSnnn,device-type,VOL=volser,SHR
// DLBL PXX,'pxx'                                  PXX data set
// EXTENT SYSnnn,volser
// EXEC DBUTLTY,SIZE=750K
         REPORT   AREA=PXX,JOBSTAT=DETAIL,SYSSTAT=SUMMARY
/*
/&
* $$ EOJ
```

Sample Report: REPORT PXX

```
Date: mm/dd/ccyy    ********************************************************************************          Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT  AREA=PXX,JOBSTAT=DETAIL,SYSSTAT=SUMMARY

                    FUNCTION=REPORT
       ❷               AREA=PXX
                       JOBSTAT=DETAIL
                       SYSSTAT=SUMMARY


       ❸
 PXX STARTED DURING MUF ENABLE              JOB=DBDVM0              MUFNAME=DBDVM0           5/06/2009  8.27.09

 PXX MUF INFORMATION UPDATE                 JOB=DBDVM0    SVC=235   MUFNAME=DBDVM01          5/06/2009  8.27.09
 PXX END, PXX CLOSED                                                                         5/06/2009  8.27.14
 PXX RESET BY CLRPXX                        JOB=DBDVM0    SVC=235   MUFNAME=DBDVM01          5/06/2009  8.32.13

 PXX END, PXX NOT CLOSED (MUF STILL ENABLED OR PXX WAS FULL AT EOJ)                           0/00/0000  0.00.00
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing are presented in this area.

The first message in the example just shown notes that the PXX was started when the MUF was enabled. This message has no SVC number, because the PXX is started before MUF determines the number to be used. It also has the same MUF name as the MUF job name, because the PXX is started before the MUF determines the MUF name to be used. After the MUF has determined this information, the second line occurs, providing the update to include the SVC and MUF name the MUF is using. The PXX END, PXX CLOSED indicates the MUF has closed the PXX during EOJ processing and at this time the PXX was not full. nt. In z/VSE the SVC is never printed, and the MUFNAME= field contains the SUBID=n value being used.

If the PXX is cleared with CLRPXX, the first message reflects PXX RESET BY CLRPXX with the current identity information. The message PXXEND, PXX NOT CLOSED (MUF STILL ENABLED OR PXX WAS FULL AT EOJ) reflects that the PXX has no close record. This occurs if MUF is still enabled or occurs if the MUF did EOJ but the PXX was full and MUF could not write the close record as shown in the following example:

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  STARTED AFTER COMM CLRPXX        JOB=DBDVMUF   SVC=209  SUBID=0                1/06/2003 7.10.15   ❹
```

❹

This page can contain information such as the following:

- A message indicating the start of the Multi-User Facility. It includes the job name, the sub-identifier, the date, and the time.

The report continues with statistics for each job that wrote statistics.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  JOB STATISTICS              CLOSING    JOB-81471-DRH402E   TASK-001             1/06/2003 16.24.01  ❶


  ❷      ❸  ❹     ❺  ❻  ❼  ❽  ❾

TABLE    URT SQBUF   UPD SYN SUB DXC GBL    REQUESTS  RETRIEVALS       ADDS     DELETES     UPDATES

BAS00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
ARA00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
FIL00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
AGR00002  0     0    Y   Y   N   Y   N            8           4          0           0           0
FLD00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
KEY00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
ELM00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
ALS00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
KWC00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
REL00002  0     0    Y   Y   N   Y   N            8           2          2           0           0
TXT00002  0     0    Y   Y   N   Y   N          125          58          3           1          24
ATZ00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
DVW00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
JOB00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
LIB00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
MEM00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
MOD00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
NOD00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
PER00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
PGM00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
PNL00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
RPT00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
STP00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
SYS00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
HSD00002  0     0    Y   Y   N   Y   N            4           4          0           0           0
PLN00002  0     0    Y   Y   N   Y   N            3           2          0           0           1
STM00002  0     0    Y   Y   N   Y   N           22          12          0           0           8
PRT00002  0     0    Y   Y   N   Y   N            0           0          0           0           0
DDD00015  0     0    Y   N   N   Y   N          616         549         14          14           0
CNO00015  0     0    Y   N   N   Y   N            2           0          0           0           0
A3300016  0     0    Y   N   N   Y   N          206         101        100           0           0
TTM00017  0     0    Y   N   N   Y   N            2           0          1           0           0

TOTALS                                         996         732        120          15          33
```

Database Job Statistics

❶

Message beginning the job statistics. It includes the job name and run unit number of the job that closed a User Requirements Table that caused the statistics to be written to the Statistics and Diagnostics Area.

❷

TABLE - The DATACOM-NAME of the table concatenated with the DATACOM-ID of the database containing the table.

❸

URT - The number used by CA Datacom/DB to keep track of the User Requirements Table. For SQL, this number is always zero.

❹

SQBUF - The number of sequential buffers.

❺

UPD - Indicates the value for UPDATE= in the User Requirements Table. UPDATE=YES means the program can update the table and can hold the table's records under exclusive control. UPDATE=NO means that there is no update capability and records being read cannot be held under exclusive control.

❻

SYN - Indicates the value for SYNONYM= in the User Requirements Table. YES indicates that the table name may be duplicated in different databases. NO specifies that there are no duplicate table names.

❼

SUB - Indicates the value for DBFLSUB= in the User Requirements Table. YES indicates the CA Datacom/DB table load subroutine DBFLSUB is being used.

❽

DXC - Indicates the value for AUTODXC= in the User Requirements Table. YES indicates that the GETIT, GETPS, GSETL, GSETP, RDUBR, RDUNE, RDUNK, RDUNR and RDUNX commands automatically drop exclusive control established by the previous command from the same Request Area. NO specifies the previous commands do not automatically drop exclusive control.

❾

GBL - Indicates the User Requirements Table parameter GETBLK= is being used to specify the buffer size used for blocked GETIT and GETPS commands.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     4
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  JOB STATISTICS              CLOSING    JOB-81471-DRH402E   TASK-001             1/06/2003 16.24.53


                                                        ❶          ❷           ❸          ❹          ❺

TABLE    URT SQBUF   UPD SYN SUB DXC GBL    REQUESTS  RETRIEVALS       ADDS     DELETES     UPDATES

BAS00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
ARA00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
FIL00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
AGR00002  0     0    Y   Y   N   Y   N             8           4          0           0           0
FLD00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
KEY00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
ELM00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
ALS00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
KWC00002  0     0    Y   Y   N   Y   N             0           0          0           0           0
REL00002  0     0    Y   Y   N   Y   N             8           2          2           0           0
TXT00002  0     0    Y   Y   N   Y   N           125          58          3           1          24
     .    .     .    .   .   .   .   .             .           .          .           .           .
     .    .     .    .   .   .   .   .             .           .          .           .           .
     .    .     .    .   .   .   .   .             .           .          .           .           .
     .    .     .    .   .   .   .   .             .           .          .           .           .
```

❶

REQUESTS - The number of total requests per table, including all CA Datacom/DB internal requests. This number includes any requests processed during transaction backout. If no requests are recorded for a table, that table is listed with zero counts.

Since internal requests are included in the total, the number of retrieval, add, delete and update requests may not equal the number shown in this column.

❷

RETRIEVALS - The number of retrieval requests recorded against this table. The count includes all requests that return data to the user (not locate commands). The commands included are: GETxx, RDUxx, REDxx, SELFR, SELNR, and SELSM. CA Datacom/DB commands and SQL statements are included in this count.

❸

ADDS - The number of add requests recorded against this table. CA Datacom/DB commands and SQL statements are included in this count.

❹

DELETES - The number of delete requests recorded against this table. CA Datacom/DB commands and SQL statements are included in this count.

❺

UPDATES - The number of update requests recorded against this table. CA Datacom/DB commands and SQL statements are included in this count.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     5
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  SUMMARY STATISTICS                                                        1/06/2003 16.27.05


REQUESTS                                      104}
SQL REQUESTS                                  102}   ❶
DATA MANAGER REQUESTS                       1,138}
I/O EVENTS - READS                            268
I/O EVENTS - WRITES                           155
EXCLUSIVE CONTROL REQUESTS                    154
EXCLUSIVE CONTROL CONFLICTS                     0
K/E BUFFER REORGANIZATION                       0
NO IXX BUFFER AVAILABLE                         0
NO DXX BUFFER AVAILABLE                         0
NO DATA BUFFER AVAILABLE                        0
NO EXPAND BUFFER AVAILABLE                      0
LAST IXX SEQUENCE NUMBER                      512
LAST DXX SEQUENCE NUMBER                      670
LAST DATA SEQUENCE NUMBER                     969
LAST EXPAND SEQUENCE NUMBER                   735
IXX BUFFER USED - 1                            16
IXX BUFFER USED - 2                            11
IXX BUFFER USED - 3                             8
IXX BUFFER USED - 4                             8
IXX BUFFER USED - 5+                          469
DXX BUFFER USED - 1                            66
DXX BUFFER USED - 2                            44
DXX BUFFER USED - 3                            29
DXX BUFFER USED - 4                            26
DXX BUFFER USED - 5+                          505
DATA BUFFER USED - 1                           95
DATA BUFFER USED - 2                           74
DATA BUFFER USED - 3                           30
DATA BUFFER USED - 4                           24
DATA BUFFER USED - 5+                         746
SEQ. READ AHEAD I/O EVENTS                      0
LOG AREA MAXIMUM PERCENT FULL                  13
SECONDARY E/C CONFLICTS                         0
TASK COMPLETION WAITS                           0
REQUEST COMPLETION WAITS                        0
SPILL COMPLETION WAITS                          0
```

Database Summary Statistics

❶

These fields report the number of requests processed:

REQUESTS

Reports the total number of service requests processed. When using the GETIT or GETPS commands in blocked mode, the total request count is the number of calls to Multi-User, not the number of detail requests for the next record.

SQL REQUESTS

The number of requests to the Multi-User Facility for SQL work. SQL requests are also included in the previously shown REQUESTS count. The difference in the number of REQUESTS reported and the number of SQL REQUESTS is equal to the number of CA Datacom/DB commands issued to the Multi-User Facility.

DATA MANAGER REQUESTS

This is the number of calls to the low-level CA Datacom/DB processor. Each CA Datacom/DB command issued to the Multi-User Facility generates one data manager request. Each SQL request generates as many data manager requests as necessary to process the SQL request. Data manager requests can also be generated (infrequently) by internal MUF processes such as restart.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     6
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  SUMMARY STATISTICS                                                          1/06/2003 16.27.05


REQUESTS                                    104
SQL REQUESTS                                102
DATA MANAGER REQUESTS                     1,138
I/O EVENTS - READS                          268  }  ❷
I/O EVENTS - WRITES                         155  }
EXCLUSIVE CONTROL REQUESTS                  154 }   ❸
EXCLUSIVE CONTROL CONFLICTS                   0 }
K/E BUFFER REORGANIZATION                     0  }  ❹
NO IXX BUFFER AVAILABLE                       0 }
NO DXX BUFFER AVAILABLE                       0 }   ❺
NO DATA BUFFER AVAILABLE                      0 }
NO EXPAND BUFFER AVAILABLE                    0 }
LAST IXX SEQUENCE NUMBER                    512  }
LAST DXX SEQUENCE NUMBER                    670  }
LAST DATA SEQUENCE NUMBER                   969  }  ❻
LAST EXPAND SEQUENCE NUMBER                 735  }
IXX BUFFER USED - 1                          16 }
IXX BUFFER USED - 2                          11 }
IXX BUFFER USED - 3                           8 }
IXX BUFFER USED - 4                           8 }
IXX BUFFER USED - 5+                        469 }
DXX BUFFER USED - 1                          66 }
DXX BUFFER USED - 2                          44 }
DXX BUFFER USED - 3                          29 }   ❼
DXX BUFFER USED - 4                          26 }
DXX BUFFER USED - 5+                        505 }
DATA BUFFER USED - 1                         95 }
DATA BUFFER USED - 2                         74 }
DATA BUFFER USED - 3                         30 }
DATA BUFFER USED - 4                         24 }
DATA BUFFER USED - 5+                       746 }
SEQ. READ AHEAD I/O EVENTS                    0
LOG AREA MAXIMUM PERCENT FULL                13
SECONDARY E/C CONFLICTS                       0
TASK COMPLETION WAITS                         0
REQUEST COMPLETION WAITS                      0
SPILL COMPLETION WAITS                        0
```

❷

The I/O EVENTS READS/WRITES fields reflect the number of physical I/O requests to the operating system (execute channel programs--EXCPs), not the number of blocks read and written (in some cases, multiple blocks may be read with a single I/O event, such as during sequential processing). The ratio of I/Os to the number of requests varies greatly depending on application requirements and design.

❸

These two fields report exclusive control activity:

REQUESTS

The number of times primary exclusive control was granted for any data record.

CONFLICTS

The number of times a task requested primary exclusive control of a record that was currently held under primary exclusive control by another task.

Exclusive control conflicts normally occur when multiple online tasks request the same records for update. The number of exclusive control conflicts compared to the number of exclusive control requests provides no clear (good/bad) analysis. A value of 1 that required a task to wait a long time can be considered bad. A value that is a high percentage of exclusive control requests can be satisfactory when the wait time was short and the heavily used application required the control.

❹

K/E BUFFER REORGANIZATION is the number of times a key/element definition had to be removed from memory to allow a different key/element definition to be read.

❺

NO ... BUFFER AVAILABLE is the number of times all of the buffers of the identified type required by a task were in use. The following are the buffer types:

IXX

Index Area buffers (high-level index blocks)

DXX

Index Area buffers (low-level index blocks)

DATA

Data area primary and secondary buffers (data area blocks)

EXPAND

Expand buffers

Note: Because of the dynamic acquisition and freeing of buffer space by CA Datacom/DB, this value cannot be maintained at zero.

❻

The LAST ... is the total number of references to a particular class of buffers.

❼

The ... BUFFER USED counters tell you how much the IXX, DXX, and data buffers allocated in the Multi-User startup options are being used and any added with the FLEXPOOL function. The following is a definition of each type of buffer:

IXX

Index Area buffers (high-level index blocks)

DXX

Index Area buffers (low-level index blocks)

DATA

Data area buffers (data area blocks)

The first time a buffer is used, after an index or data block is read into a block of memory, the first counter is set to 1. A second use of the same block, if it is still in memory, increments the second counter to 2. The third and fourth uses increment the third and fourth counters to 3 and 4, respectively. Fifth and subsequent uses of the block increment the fifth counter to 5+. No counter is incremented if a block is not used, such as after an I/O error, or if a block is read-ahead sequentially but the application terminated the sequential process prior to end-of-file.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     7
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  SUMMARY STATISTICS                                                          1/06/2003 16.27.05


REQUESTS                                    104
SQL REQUESTS                                102
DATA MANAGER REQUESTS                     1,138
I/O EVENTS - READS                          268
I/O EVENTS - WRITES                         155
EXCLUSIVE CONTROL REQUESTS                  154
EXCLUSIVE CONTROL CONFLICTS                   0
K/E BUFFER REORGANIZATION                     0
NO IXX BUFFER AVAILABLE                       0
NO DXX BUFFER AVAILABLE                       0
NO DATA BUFFER AVAILABLE                      0
NO EXPAND BUFFER AVAILABLE                    0
LAST IXX SEQUENCE NUMBER                    512
LAST DXX SEQUENCE NUMBER                    670
LAST DATA SEQUENCE NUMBER                   969
LAST EXPAND SEQUENCE NUMBER                 735
IXX BUFFER USED - 1                          16
IXX BUFFER USED - 2                          11
IXX BUFFER USED - 3                           8
IXX BUFFER USED - 4                           8
IXX BUFFER USED - 5+                        469
DXX BUFFER USED - 1                          66
DXX BUFFER USED - 2                          44
DXX BUFFER USED - 3                          29
DXX BUFFER USED - 4                          26
DXX BUFFER USED - 5+                        505
DATA BUFFER USED - 1                         95
DATA BUFFER USED - 2                         74
DATA BUFFER USED - 3                         30
DATA BUFFER USED - 4                         24
DATA BUFFER USED - 5+                       746
SEQ. READ AHEAD I/O EVENTS                    0   ❶
LOG AREA MAXIMUM PERCENT FULL                13   ❷
SECONDARY E/C CONFLICTS                       0   ❸
TASK COMPLETION WAITS                         0 } ❹
REQUEST COMPLETION WAITS                      0 }
SPILL COMPLETION WAITS                        0 }
```

❶

SEQ. READ AHEAD I/O EVENTS is the number of read-ahead I/O events which occurred during the execution of GETIT and GETPS commands.

This number indicates how effective these commands are in batch jobs. The number of blocks per I/O is generally equal to one-half the number of SEQBUFS= provided in the User Requirements Table. The next low-level index block is read ahead (when using the GETIT command).

❷

LOG AREA MAXIMUM PERCENT FULL is the highest percent full the Log Area reached.

❸

SECONDARY E/C CONFLICTS is the number of times there was a conflict of exclusive control at the secondary level caused by jobs with transaction backout specified. Once such jobs update a record, CA Datacom/DB holds the record under secondary exclusive control until a checkpoint is issued. This allows access to the record, if CA Datacom/DB needs to invoke transaction backout.

❹

... COMPLETION WAITS reports the number of times requests had to wait for service. Continues on the next page of the report.

TASK COMPLETION WAITS

Indicates the number of times one task had to wait on the completion of another task because of transaction backout contention. This happens only when there are secondary exclusive control conflicts. It is therefore related to the previously shown secondary exclusive control statistic and may show high contention for individual rows by applications.

REQUEST COMPLETION WAITS

The number of times a request had to wait either to log a request because the Log Area was full and the oldest log block was not spillable (due to a long-running maintenance request) or for the index queue processing to complete prior to closing the index data set.

SPILL COMPLETION WAITS

The number of times one task had to wait on the completion of a Log Area spill before being able to log a record and continue. This statistic could signify a Log Area which is too small.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     8
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  (CONTINUED)

ACCOUNTING COMPLETION WAITS                    0 }  ❶
SECURITY COMPLETION WAITS                      0 }
INDEX QUEUE PROCESSING                         0  } ❷
INDEX QUEUE OVERFLOWS                          0  }
INDEX BLOCK SPLITS - IXX                       0 }
INDEX BLOCK SPLITS - DXX                       0 }  ❸
INDEX BLOCK DELETES                            0 }
CBSBFR SPILL COUNT                             0  }
CBSBFR ENTRIES AVAILABLE                      53  }
MOST CBS ENTRIES USED                          0  }
SETS PROCESSED                                 1  } ❹
SETS REQUIRING TEMPORARY INDEX                 0  }
TEMPORARY INDEX ENTRIES BUILT                  0  }
CBS MEMORY ALLOCATIONS                         0  }
WRITEPEND WRITE IO                             0 }  ❺
DATASECURE KEY DELETES                         0  } ❻
DATASECURE LOOP DELETES                        0  }
RE-EXPAND COMPRESSED RECORDS                   1
LOGPOOL BUFFER WAITS                           0
INDIRECT I/O WAITS                         2,551
LOG CONTROL BLOCK WRITE I/O                  139
LOG BUFFER FULL WRITE I/O                  3,784
LOG COMMAND WRITE I/O                        932
LOG OTHER WRITE I/O                        1,203
LOGPOOL WRITE I/O                            141
LOG 2-PHASE COMMIT WRITE I/0                   3
DATA2 BUFFER USED - 1                         95
DATA2 BUFFER USED - 2                         74
DATA2 BUFFER USED - 3                         30
DATA2 BUFFER USED - 4                         24
DATA2 BUFFER USED - 5+                       746

REQUEST PENDING - 01                       2,220
REQUEST PENDING - 02                         331
REQUEST PENDING - 03                          27
```

❶

... COMPLETION WAITS reports the number of times requests had to wait for service.

ACCOUNTING COMPLETION WAITS

The number of times that the CA Datacom/DB Accounting Facility tried to add information to the accounting tables but the buffer was full or closed and the task had to wait. If this number is large, review the accounting tables defined and the accounting parameters specified in the Multi-User startup options.

SECURITY COMPLETION WAITS

The number of times that security had to wait for system information.

❷

These two fields report index queue processing:

INDEX QUEUE PROCESSING

The number of index queue entries processed. Queue processing includes:

- All high-level block processing
- Block deletions of low-level blocks

This statistic is useful primarily to Technical Support for debugging purposes.

INDEX QUEUE OVERFLOWS

The number of times the index queue overflowed. If an entry does not fit in the queue, it is ignored and attempted later. This statistic is useful primarily to Technical Support for debugging purposes.

❸

INDEX BLOCK SPLITS - reports the number of times a high-level (IXX) or low-level (DXX) index block was split into two blocks and the number of times an index block, both high and low levels, was logically deleted. These statistics are useful primarily to Technical Support for debugging purposes.

❹

The following fields report Compound Boolean Selection processing:

CBSBFR SPILL COUNT

The number of times the Compound Boolean Selection buffer became full, causing CA Datacom/DB to store active set definitions as index entries with a special key ID. The buffer in which active set definitions are stored is defined by the Multi-User startup option CBS.

CBSBFR ENTRIES AVAILABLE

The number of 185-byte entries available in the Compound Boolean Selection buffer (defined by the Multi-User startup option CBS value) for storing active set definitions. A set definition typically requires one to three entries.

MOST CBS ENTRIES USED

The maximum number of Compound Boolean Selection buffer entries used to store active set definitions. If this number is equal to the number of available entries, the previously shown CBSBFR SPILL COUNT value indicates how often a set definition index entry was accessed.

SETS PROCESSED

The total number of sets processed.

SETS REQUIRING TEMPORARY INDEX

The number of sets processed that built a temporary index.

TEMPORARY INDEX ENTRIES BUILT

The number of temporary index entries built to satisfy SELFR command processing.

CBS MEMORY ALLOCATIONS

The number of times that the Compound Boolean Selection Facility had to dynamically allocate the RWTSA due to the complexity of the requests and the number of keys being used.

❺

WRITEPEND WRITE IO is the number of logging (physical) writes (controlled by the Multi-User startup option LOGPEND) that occurred during this cycle.

❻

DATASECURE ... these fields are reserved for future use.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     9
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  (CONTINUED)

ACCOUNTING COMPLETION WAITS                    0
SECURITY COMPLETION WAITS                      0
INDEX QUEUE PROCESSING                         0
INDEX QUEUE OVERFLOWS                          0
INDEX BLOCK SPLITS - IXX                       0
INDEX BLOCK SPLITS - DXX                       0
INDEX BLOCK DELETES                            0
CBSBFR SPILL COUNT                             0
CBSBFR ENTRIES AVAILABLE                      53
MOST CBS ENTRIES USED                          0
SETS PROCESSED                                 1
SETS REQUIRING TEMPORARY INDEX                 0
TEMPORARY INDEX ENTRIES BUILT                  0
CBS MEMORY ALLOCATIONS                         0
WRITEPEND WRITE IO                             0
DATASECURE KEY DELETES                         0
DATASECURE LOOP DELETES                        0
RE-EXPAND COMPRESSED RECORDS                   1  }
LOGPOOL BUFFER WAITS                           0  }
INDIRECT I/O WAITS                         2,551  }
LOG CONTROL BLOCK WRITE I/O                  139  }
LOG BUFFER FULL WRITE I/O                  3,784  }
LOG COMMAND WRITE I/O                        932  }
LOG OTHER WRITE I/O                        1,203  }
LOGPOOL WRITE I/O                            141  }  ❼
LOG 2-PHASE COMMIT WRITE I/O                   3  }
DATA2 BUFFER USED - 1                         95  }
DATA2 BUFFER USED - 2                         74  }
DATA2 BUFFER USED - 3                         30  }
DATA2 BUFFER USED - 4                         24  }
DATA2 BUFFER USED - 5+                       746  }
                                                  }
REQUEST PENDING - 01                       2,220  }
REQUEST PENDING - 02                         331  }
REQUEST PENDING - 03                          27  }
```

❼

These fields report the following:

RE-EXPAND COMPRESSED RECORDS

As compressed records are read, they are expanded into an expand buffer. If a record is updated or deleted, the same buffer is found or the record is re-expanded. Priority of expand buffers is to preserve records under exclusive control. This parameter is provided to allow tuning of the number of expand buffers.

LOGPOOL BUFFER WAITS

This is the count of times that transaction backout was running, attempting to read a log block, but no LOGPOOL buffer was available. The number of log buffers for transaction backout is provided during Multi-User Facility startup. If multiple transaction backouts occur concurrently, contention can occur if fewer LOGPOOL buffers exist than tasks. Because multiple transaction backouts only rarely occur together, trying to keep this stat at zero would, in general, misuse memory, but be aware that when it does occur transaction backout is slowed.

INDIRECT I/O WAITS

An indirect I/O wait is the condition where a task needs a block which is being read/written. The current task did not start the I/O with wait, but must wait on its completion. The current task could have started the I/O without wait such as reading ahead. These counts are also included in REQUEST PENDING.

LOG CONTROL BLOCK WRITE I/O

This is the number of times the Log control block was written. The control block information is required for RESTART. This counter is part of the information to provide details on the LXX I/O.

LOG BUFFER FULL WRITE I/O

This is the number of times that a log buffer was written because it was full.

LOG COMMAND WRITE I/O

This is the number of times the Log buffer containing logged data was written because of a command. Commands which cause transaction boundaries (COMIT, LOGCP, LOGCR, ROLBK, CLOSE) are always written. Maintenance commands ADDIT, DELET, and UPDAT are only written if the table definition indicated writes, the URT indicates writes, or the Multi-User startup option LOGPEND indicated no pipelining. Commands which are optional are LOGIT and LOGDW. Some internal system commands may also write the log block prior to full.

Comparing the ratio of COMMAND WRITE I/O to BUFFER FULL WRITE I/O can be used to attempt tuning by reducing the number of LXX writes.

LOG OTHER WRITE I/O

Indicates the times a non-full log block had to be written to support RESTART when an index or data block had to be written.

LOGPOOL WRITE I/O

This represents the number of write I/O to the LOGPOOL buffers used by transaction backout.

LOG 2-PHASE COMMIT WRITE I/0

This represents the number of write I/O to the LOGPN buffers used by two-phase commit to manage system failure between a coordinating site and one or more remote sites.

DATA2 BUFFER USED - 1 through 5+

These counters tell you how much the secondary data area buffers allocated in the Multi-User startup options are being used and any added with the FLEXPOOL function. See the IXX, DXX, and DATA BUFFER USED fields for an explanation of how these counters are used.

REQUEST PENDING

The number of non-I/O waits that occurred during this cycle. The number following the dash indicates the number of tasks that had to wait. A non-I/O wait count means:

- A buffer was not available.
- An exclusive control conflict was encountered.
- An I/O event requested by one task caused another task to wait.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:    10
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  SUMMARY STATISTICS                                                          1/06/2003 16.27.05


                          ❶                ❷                  ❸                ❹                ❺

TABLE/AREA          REQUESTS   I/O EVENTS - READS  I/O EVENTS - WRITES       LOGICAL READS      LOGICAL WRITES

CXX                        0                  88                   36                 753                 129
LXX                        0                   3                   37                   0                 201
AGR00002                   8                   3                    0                   6                   0
ALS00002                   0                   1                    0                   2                   0
ARA00002                   0                   1                    0                   2                   0
ATZ00002                   0                   1                    0                   2                   0
BAS00002                   0                   1                    0                   2                   0
DVW00002                   0                   1                    0                   2                   0
ELM00002                   0                   1                    0                   2                   0
FIL00002                   0                   1                    0                   2                   0
FLD00002                   0                   1                    0                   2                   0
HSD00002                   4                   5                    0                   6                   0
IXX00002                   0                  41                   24                 304                  53
JOB00002                   0                   1                    0                   2                   0
KEY00002                   0                   1                    0                   2                   0
KWC00002                   0                   1                    0                   2                   0
  .                        .                   .                    .                   .                   .
  .                        .                   .                    .                   .                   .
  .                        .                   .                    .                   .                   .

STP00002                   0                   1                    0                   2                   0
SYS00002                   0                   1                    0                   2                   0
TXT00002                 125                  11                    9                  86                  36
IXX00006                   0                   2                    1                   3                   1
CNO00015                   2                   0                    0                   0                   0
DDD00015                 616                  31                   11                 579                  37
IXX00015                   0                  30                   20                 247                 103
SIT00015                   0                   1                    0                   2                   0
A3300016                 206                   0                    0                   0                   0
IXX00016                   0                   5                    3                 112                 102
SQ100016                   0                  10                    3                 211                 102
IXX00017                   0                   4                    1                   8                   1
TTM00017                   2                   1                    0                   2                   0
TOTAL                    996                 268                  151               2,396                 781
```

These columns provide a breakdown of the totals given for REQUESTS, I/O EVENTS - READS, and I/O EVENTS - WRITES printed at the beginning of the report. Use this information in conjunction with the Accounting Facility to pinpoint inefficiencies by determining what kind of requests are occurring.

You can use the ratio of LOGICAL READS to I/O EVENTS - READS and LOGICAL WRITES to I/O EVENTS - WRITES as an indicator of the efficiency of your buffer size. The greater the number of logical reads or writes to I/O events, the better your buffer use is and the more effective is your pipeline.

❶

REQUESTS - The number of requests per table which have occurred and been recorded in the Statistics and Diagnostics Area.

❷

I/O EVENTS - READS - The number of I/O reads which have occurred and been recorded in the Statistics and Diagnostics Area.

❸

I/O EVENTS - WRITES - The number of I/O writes which have occurred and been recorded in the Statistics and Diagnostics Area.

❹

LOGICAL READS - The number of internal read requests which have occurred and been recorded in the Statistics and Diagnostics Area.

❺

LOGICAL WRITES - The number of internal write requests which have occurred and been recorded in the Statistics and Diagnostics Area.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:    11
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                                General Utility                               *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DATA BASE  SUMMARY STATISTICS                                                          1/06/2003 16.27.05


TABLE           RETRIEVALS                ADDS              DELETES             UPDATES

AGR00002                 4                   0                    0                   0
HSD00002                 4                   0                    0                   0
PLN00002                 2                   0                    0                   1
REL00002                 2                   2                    0                   0
STM00002                12                   0                    0                   8
TXT00002                58                   3                    1                  24
CNO00015                 0                   0                    0                   0
DDD00015               549                  14                   14                   0
A3300016               101                 100                    0                   0
TTM00017                 0                   1                    0                   0

TOTALS                 732                 120                   15                  33
```

Database Summary Requests Totals

The column headings refer to the following:

RETRIEVALS

The total number of retrieval requests since Multi-User enable. This count includes all requests that return data to the user. If no requests are recorded for a table, that table is not listed.

ADDS

The total number of add requests since Multi-User enable.

DELETES

The total number of delete requests since Multi-User enable.

UPDATES

The total number of update requests since Multi-User enable.

Expression Dump

Value-level operations are represented internally in an "expression tree" data structure. Expression trees may appear in the Statistics and Diagnostics Area (PXX) DUMPS=FULL or DUMPS=TRACE Reports when traces are turned on, or to provide additional information for certain SQLCODE errors (also see Dumping SQL Internal Control Structures (see page 608)). For example, an attempt to compare a string and a number causes the predicate to be dumped.

The expression tree dump is printed in the following format. The dump is provided primarily for Technical Support to assist in debugging a problem.

```
  LVL 1 NODE hhhhhhhh OP=hhhhhhhh NXT=hhhhhhhh$L¢ QCB=hhhhhhhh VAL=hhhhhhhh
NAM=hhhhhhhh
  $... . PROCTYPE = hhhh  STEP=nn  OPER = nnn xxxxxxxx¢
  ... . tblTyp (#nnnn:nnnn.nnnn): dataType $NOT NULL¢ $POSITIVE ONLY¢
  $... . NAME: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx¢
  $... . value¢
  $... . CONTENTS=¢
    LVL 2
      LVL n
```

Line 1

This line indicates the position of the node in the tree structure and various addresses, which are all in hex. The following are the headings and descriptions.

LVL

Node level within tree displayed. (Sometimes only a sub-tree of a larger tree is displayed.)

NODE

Address of node.

OP

Address of first operand. Zero indicates a leaf node.

NXT

Address of next node at same level or, if suffixed with L, address of result (next higher) node. Indicates the node is the last (or only) node at the top level. When only a sub-tree is being displayed, NXT may be temporarily set to zero.

QCB

Address of Query Control Block.

VAL

Address of value.

NAM

Address of name. Name is not available in all cases.

Line 2

This line does not appear for leaf nodes. In other words, it only appears if the node is the result of an operation. The following are the headings and descriptions.

PROCTYPE

Internal bit values that indicate the process class, (such as predicate, scalar function), or attribute of value (such as descending ORDER BY column).

STEP

Step in which value is computed or made available. Not applicable for all node types.

OPER

Operation performed by node. The number is a subscript into a function pointer table. It is followed by an abbreviation of the function name.

Line 3

The following are the headings and descriptions.

tblType

Indicates type of table:

CNSCONV

Constraint

COLCONV

Result of a column data type conversion

HOSTOUT

Output host variable

HOSTVAR

Input host variable

KEY DEF

Key defintion

LITERAL

Literal

RESULT

Result of a value-level operation, such as the result of an expression, function, or data type conversion

QCB PTR

Indicates VALUE is a Query Control Block

SPLREG

Special register

TBL nn

Reference to column in table, where "nn" refers to the table's position in the FROM clause, relative to zero

#nnnn

Column number.

nnnn.nnnn

Offset in the row and the length in number of bytes.

dataType

Data type, using CREATE TABLE syntax.

NOT NULL

Indicates value does not have a null indicator.

POSITIVE ONLY

Indicates column reference is a numeric data type with only positive signed values. (CA Datacom/DB FIELD entity-occurrence attribute TYPE-NUMERIC=P.)

Line 4 (NAME:)

Name of value, if any. NAME may not be available if dump is at execution-time.

Line 5 (value)

This line appears when the node's value is one of the following special types. The following are the headings and descriptions.

KEY DEFINITION

Hex dump of key definition in internal format.

COLUMN NAME =

Unresolved column reference.

COLUMN REFERENCE:

"authID.tableName.columnName" of unresolved column reference.

Line 6 (CONTENTS=)

This line appears when the node's value is not one of the special cases shown in line 5. The following are the headings and descriptions.

CONTENTS=NULL

NULL value.

EXPR VALUE CONTENTS

Value dumped in hex format.

string value

Only the first 48 bytes are displayed. Value is terminated by a period (so the number of trailing blanks can be determined).

numeric value

Data types NUMERIC and DECIMAL are dumped in hex format.

date/time value

The date and time displayed in ISO format.

Sample Report

The following is an example of an "expression tree" dump when an SQLCODE -41 (incompatible data types) was received.

```
DATA BASE  SQL DUMP SUMMARY                                    1/06/2003
12.21.56


ERROR DETECTED IN PROGRAM  EXPRCONV   AT LINE  367
ERROR MESSAGE TEXT:  DATATYPES OF OPERATION '>' NOT COMPATIBLE

SQL ERROR CODE IS:     -41
PLAN NAME IS:          ISDBXXXX45291775
AUTHORIZATION ID IS:        WILRI01
STATEMENT ID IS:       0
DSF RETURN CODE IS:    BLANKS
CA-STAR RETURN CODE IS:
DB RETURN CODE IS:       (0)
DB COMMAND IS:         QPRPD
                                                    1/06/2003 12.21.56
LVL 1 NODE 047B3340 OP=047B32D2 NXT=00000000L QCB=047B3168 VAL=00000000 NAM=00000000
... . PROCTYPE = 0x8008  STEP=-01  OPER = 249 PREDPREP
... . RESULT  (#0000:00000.00000): UNKNOWN
  LVL 2 NODE 047B32D2 OP=00000000 NXT=047B330C    QCB=047B3168 VAL=0476F058 NAM=047C7B46
  ... . TBL 00 (#0000:00000.00008): CHAR (8) NOT NULL
  ... . NAME: COL1 >  COL4              .
  LVL 2 NODE 047B330C OP=00000000 NXT=047B3340L QCB=047B3168 VAL=0476F0B0 NAM=047C7B4E
  ... . TBL 00 (#0003:00015.00006): NUMERIC (5,0)
  ... . NAME: COL4                  .
```

Reporting on the Compound Boolean Selection Facility

The Compound Boolean Selection Facility Diagnostic Report provides information that can pinpoint the efficiency of SELFR commands. The DUMPS=TRACE parameter causes this information, which is stored in the Statistics and Diagnostics Area, to print.

This report can be used to determine why a SELFR request is executing slowly. The report identifies the source program statement, lists the request (the "compiled" version if using CA Ideal) and key definitions, and information about each of the following four phases of the SELFR command, indicating what resources were used and why:

- Optimization
- Ordering
- Retrieval
- Selection

This report includes two examples. The fields that appear on these reports depend on the specific CBS requests, therefore, your report fields may not be included in these examples. For a description of each possible field in this report, see the discussion of the Compound Boolean Selection Facility Diagnostic report in the *CA Datacom/DB Database and System Administration Guide*.

Example JCL

The following shows the command to print the Compound Boolean Selection Facility Diagnostic Report.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL PXX,'pxx'                                  PXX data set
 // EXTENT SYSnn,volser
 // EXEC DBUTLTY,SIZE=750K
        REPORT   AREA=PXX,DUMPS=TRACE
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT PXX - CBS Facility Diagnostic Report

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT AREA=PXX,DUMPS=TRACE

                    FUNCTION=REPORT
       ❷               AREA=PXX
                       DUMPS=TRACE


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing are presented in this area.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

*** CMD= SELFR DBID= 001   TABLE= POL USER= $$$  $ID.CBSTS001(001) 000102    DATE= 1062003 TIME= 145837   ❹
********** REQUEST SPECIFICATION **************
DATAVIEW ELEMENT: PO    OFFSET IN RECORD: 00000 LENGTH: 00005
DATAVIEW ELEMENT: LI    OFFSET IN RECORD: 00005 LENGTH: 00003
DATAVIEW ELEMENT: PN    OFFSET IN RECORD: 00008 LENGTH: 00003
DATAVIEW ELEMENT: QTY   OFFSET IN RECORD: 00011 LENGTH: 00003    ❺
DATAVIEW ELEMENT: UCOST OFFSET IN RECORD: 00014 LENGTH: 00005
WHERE:   EQ Z N PO    0000 0005 V 0005 12345 F1F2F3F4F5
WHERE: A EQ C N PN    0000 0003 V 0003 XYZ E7E8E9
KEY DEF: POLI  ( 000B )          TYPE: NATIVE   ( D0 ) ROWS: 00002974
   FIELDS= 00000.005   ; 00005.003   ;
   CARD  = 000000010   ; 000000200
   STATS: 012C 0000 00CC 0000 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
KEY DEF: PN    ( 000F )          TYPE: KEYINC=Y ( 40 ) ROWS: 00002974
   FIELDS= 00008.003   ;
   CARD  = 000000100   ;
   STATS: 012C 0000 00CC 0000 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
PRI. KEY PN
        FROM= XYZ
    (IN HEX)= E7E8E9
          TO= XYZ
    (IN HEX)= E7E8E9
SEC. KEY POLI
        FROM= 12345
    (IN HEX)= F1F2F3F4F5000000
          TO= 12345
    (IN HEX)= F1F2F3F4F5FFFFFF
*************** OPTIMIZATION ******************
CBSOR REASONS: 1-5: NNNNN 6: D 7-9: YNY 10-17: YYNYNNYN 18-21: NNNN
TYPE OPTIMIZATION......: POPULATION DEPENDENT
INDEX BUFFER REFERENCES: 00000013
*************** SELECTION *********************
TYPE SELECTION.........: *INDIRECT*  DUE TO:
     O KEY INTERSECTION
     O KEY NOT AVAILABLE
*************** ORDERING **********************
TEMPORARY INDEX NOT REQUIRED.
*************** RETRIEVAL *********************
DATA RECORDS ACCESSED DUE TO: SELECTION
***END OF SELFR CMD FOR SET  00000001
CBS SET NUMBER  00000001  BEING RELEASED
TOTAL NBR INDEX ENTRIES SCANNED 00000013
TOTAL NBR DATA AREA ROWS READ   00000001
TOTAL NBR ROWS ACCEPTED         00000001
***END-OF-SET***
```

Example 1

❹

Request is made by CA Ideal program CBSTS001, version 001, system $ID, on line 000102.

❺

The REQUEST SPECIFICATION section displays the following information:

DATAVIEW ELEMENT:

The program requests a DATAVIEW with elements PO, LI, PN, QTY and UCOST for table POL. The element name is followed by its offset in the record (not including RCE) and its length. The first element is at offset 0.

WHERE:

Displays the contents of the Selection Predicate from the Request Qualification Area. In this example, rows where PO=12345 and PN=XYZ are requested. For an explanation of the remaining fields, see the *CA Datacom/DB Database and System Administration Guide*.

KEY DEF:

Displays information for the keys defined in table POL. This group of fields is repeated for each key.

Keys POLI and PN are followed by the key ID (000B) and (000F).

TYPE:

For the POLI key, the type is NATIVE. For the PN key, the type is KEYINC=Y. Codes representing the key attributes, which are for internal use, are printed in parentheses after the key type.

ROWS:

The number of rows indexed by this key. The number is initialized by the LOAD or RETIX function and is maintained by adds and deletes to the table.

FIELDS=

Field PO starts at offset zero in the record for a length of 5. The second field of the POLI key follows the semicolon for an offset of 5 and a length of 3. Field PN references the first and only field of key PN which is at offset 8 in the record for a length of 3.

CARD =

Displays the cardinalities for the columns in the index. For example, if the key contains the branch and dept columns and the cardinalities displayed are 10 and 50, this means there are 10 branches and 50 combinations of branches and departments.

Cardinality is important in understanding the selection efficiency of an index. To continue this example, if there are 1,000 rows in the table and the values are evenly distributed, the search condition "WHERE BRANCH = x" would search 1,000 divided by the cardinality of 10 which is 100 rows. If "AND DEPT = y" is added, then only 1,000 divided by 50, or 20 rows, would be searched

Cardinality is initialized by the LOAD or RETIX function. It is not updated by normal table maintenance requests. Therefore, optimum selection efficiency could require using RETIX to keep this statistic current.

STATS:

Statistical population counting for internal use only.

PRI. KEY

Displays the name of the primary key used in the search and the range of key values in both character and hexadecimal. In this example: the optimizer selected key PN as primary traversal key with the range of key character values xyz to xyz and hexadecimal values E7E8E9 to E7E8E9.

SEC. KEY

Displays the name of the secondary key used in the search and the range of key values in both character and hexadecimal. In this example: it also selected key POLI as a secondary traversal key with the range of key character values 12345 to 12345 and hexadecimal values F1F2F3F4F5000000 to F1F2F3F4F5FFFFFF.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

*** CMD= SELFR DBID= 001   TABLE= POL USER= $$$  $ID.CBSTS001(001) 000102    DATE= 1062003 TIME= 145837
********** REQUEST SPECIFICATION **************
DATAVIEW ELEMENT: PO    OFFSET IN RECORD: 00000 LENGTH: 00005
DATAVIEW ELEMENT: LI    OFFSET IN RECORD: 00005 LENGTH: 00003
DATAVIEW ELEMENT: PN    OFFSET IN RECORD: 00008 LENGTH: 00003
DATAVIEW ELEMENT: QTY   OFFSET IN RECORD: 00011 LENGTH: 00003
DATAVIEW ELEMENT: UCOST OFFSET IN RECORD: 00014 LENGTH: 00005
WHERE:   EQ Z N PO    0000 0005 V 0005 12345 F1F2F3F4F5
WHERE: A EQ C N PN    0000 0003 V 0003 XYZ E7E8E9
KEY DEF: POLI  ( 000B )          TYPE: NATIVE   ( D0 ) ROWS: 00002974
   FIELDS= 00000.005   ; 00005.003   ;
   CARD  = 000000010   ; 000000200
   STATS: 012C 0000 00CC 0000 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
KEY DEF: PN    ( 000F )          TYPE: KEYINC=Y ( 40 ) ROWS: 00002974
   FIELDS= 00008.003   ;
   CARD  = 000000100   ;
   STATS: 012C 0000 00CC 0000 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
PRI. KEY PN
        FROM= XYZ
    (IN HEX)= E7E8E9
          TO= XYZ
    (IN HEX)= E7E8E9
SEC. KEY POLI
        FROM= 12345
    (IN HEX)= F1F2F3F4F5000000
          TO= 12345
    (IN HEX)= F1F2F3F4F5FFFFFF
**************** OPTIMIZATION ******************
CBSOR REASONS: 1-5: NNNNN 6: D 7-9: YNY 10-17: YYNYNNYN 18-21: NNNN
TYPE OPTIMIZATION......: POPULATION DEPENDENT                          ❻
INDEX BUFFER REFERENCES: 00000013
**************** SELECTION **********************
TYPE SELECTION.........: *INDIRECT*  DUE TO:         ❼
     0 KEY INTERSECTION
     0 KEY NOT AVAILABLE
**************** ORDERING ***********************
TEMPORARY INDEX NOT REQUIRED.                       ❽
*************** RETRIEVAL **********************
DATA RECORDS ACCESSED DUE TO: SELECTION
***END OF SELFR CMD FOR SET  00000001
CBS SET NUMBER  00000001  BEING RELEASED
TOTAL NBR INDEX ENTRIES SCANNED 00000013     ❾
TOTAL NBR DATA AREA ROWS READ   00000001
TOTAL NBR ROWS ACCEPTED         00000001
***END-OF-SET***
```

❻

The OPTIMIZATION section displays the following information:

CBSOR REASONS:

See the *CA Datacom/DB Database and System Administration Guide* to interpret CBSOR values for 1-5, 6, 7-9, 10-17, and 18-21.

TYPE OPTIMIZATION:

The choice of which key to use is POPULATION DEPENDENT.

INDEX BUFFER REFERENCES:

Indicates the type of optimization.

This example requires 13 index buffer references to determine that the previously shown key PN is best.

❼

The SELECTION section displays the following information at TYPE SELECTION: Selection is INDIRECT because no *single* key can be used to directly locate qualifying rows (KEY NOT AVAILABLE), even though KEY INTERSECTION is used.

❽

The ORDERING section displays information about use of a temporary index.

TEMPORARY INDEX NOT REQUIRED. For an explanation of the reasons that may be listed if a temporary index **is** required, see the *CA Datacom/DB Database and System Administration Guide*.

❾

The RETRIEVAL section displays the following information:

DATA RECORDS ACCESSED DUE TO: SELECTION

Indicates rows were retrieved due to SELECTION criteria.

*****END OF SELFR CMD FOR SET 00000001**

SELFR processing has completed for the designated set.

CBS SET NUMBER 00000001 BEING RELEASED

An explicit (SELPR) or implicit request to release the set was processed.

TOTAL NBR INDEX ENTRIES SCANNED

Number of index entries read in order to satisfy the SELxx requests for set.

TOTAL NBR DATA AREA ROWS READ

Number of data area rows read in order to satisfy the SELxx requests for set.

TOTAL NBR ROWS ACCEPTED

Number of rows returned to requestor.

*****END-OF-SET*****

No more diagnostics for this set.

If this is a frequent request, defining a key containing both PO and PN would eliminate statistical population counting and index scanning since TYPE SELECTION would then be DIRECT.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     4
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

 ** CMD= SELFR DBID= 001   TABLE= PMF USER= $ID1 $ID.CBSTS005(001) 000102    DATE= 1062003 TIME= 105417   ❶
********** REQUEST SPECIFICATION **************
DATAVIEW ELEMENT: EMDTA OFFSET IN RECORD: 00000 LENGTH: 00080   ❷
WHERE:   LT Z N EMDTA 0000 0005 V 0005 00375 F0F0F3F7F5
ORDER-BY: D E EMDTA 0075 0005 P Y
KEY DEF: EMPNO ( 0001 )          TYPE: NATIVE   ( D0 ) ROWS: 00005333
   FIELDS= 00000.005   ;
   CARD  = 000000200   ;
   STATS: 006E 0015 005C 000C 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
KEY DEF: STZIP ( 0002 )          TYPE: KEYINC=Y ( 40 ) ROWS: 00005333
   FIELDS= 00068.002   ; 00070.005   ;
   CARD  = 000000060   ; 000000200
   STATS: 012C 0000 00CC 0000 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
PRI. KEY EMPNO
        FROM=
    (IN HEX)= 0000000000
          TO= 00374
    (IN HEX)= F0F0F3F7F4
INDX WRK TBL SIZE (IN HEX)............... 00003C70
**************** OPTIMIZATION ******************
CBSOR REASONS: 1-5: NYYNN 6: I 7-9: NYN 10-17: NYNNNNNN 18-21: YNYN
TYPE OPTIMIZATION......: POPULATION INDEPENDENT
INDEX BUFFER REFERENCES: 00000000
**************** SELECTION *********************
TYPE SELECTION.........: *DIRECT*
**************** ORDERING **********************
TEMPORARY INDEX REQUIRED DUE TO:
     O ORDER-BY FIELDS
     O SIGNED NUMERIC FIELDS
ENTRIES INDEXED........: 00000200
LOGIO..................: 00000609
**************** RETRIEVAL *********************
RETRIEVAL FROM TEMP INDEX NOT USED DUE TO DATA NOT AVAIL.
DATA RECORDS ACCESSED DUE TO: RETRIEVAL
***END OF SELFR CMD FOR SET  00000007
CBS SET NUMBER  00000007  BEING RELEASED
TOTAL NBR INDEX ENTRIES SCANNED 00000400
TOTAL NBR DATA AREA ROWS READ   00000400
TOTAL NBR ROWS ACCEPTED         00000200
***END-OF-SET***
```

Example 2

❶

The request is made by CA Ideal program CBSTS005, Version 001, system $ID, on line 000102.

❷

The REQUEST SPECIFICATION section displays the following information:

DATAVIEW ELEMENT:

The program requests a DATAVIEW with element EMDTA for table PMF. The element name is followed by its offset in the record (not including RCE) and its length. The first element is at offset 0.

WHERE:

Displays the contents of the Selection Predicate from the Request Qualification Area. In this example, rows where the first 5 bytes of element EMDTA, an unsigned zone-decimal field, is LT 00375. For an explanation of the remaining fields, see the *CA Datacom/DB Database and System Administration Guide*.

ORDER-BY:

Rows are to be returned ordered by the signed packed-decimal field starting at offset 75 of element EMDTA for a length of 5 bytes. For an explanation of the way the fields are displayed, see the *CA Datacom/DB Database and System Administration Guide*.

KEY DEF:

This row displays information about the keys defined for table PMF. This group of fields is repeated for each key traversed. Keys EMPNO and STZIP are followed by the key IDs (0001) and (0002).

TYPE:

For the EMPNO key, the type is NATIVE. For the STZIP key, the type is KEYINC=Y. Codes representing the key attributes, which are for internal use, are printed in parentheses after the key type.

ROWS:

The number of rows indexed by this key. The number is initialized by the LOAD or RETIX function and is maintained by adds and deletes to the table.

FIELDS=

The only field of the key EMPNO starts at offset zero in the record for a length of 005. The first field in the STZIP key starts at offset 68 for a length of 2 and the second field in the key starts at offset 70 for a length of 5.

CARD =

Displays the cardinalities for the columns in the index. For example, if the key contains the branch and dept columns and the cardinalities displayed are 10 and 50, this means there are 10 branches and 50 combinations of branches and departments.

Cardinality is important in understanding the selection efficiency of an index. To continue this example, if there are 1,000 rows in the table and the values are evenly distributed, the search condition "WHERE BRANCH = x" would search 1,000 divided by the cardinality of 10 which is 100 rows. If "AND DEPT = y" is added, then only 1,000 divided by 50, or 20 rows, would be searched.

Cardinality is initialized by the LOAD or RETIX function. It is not updated by normal table maintenance requests. Therefore, optimum selection efficiency could require using RETIX to keep this statistic current.

STATS:

Statistical population counting for internal use only.

PRI. KEY

Displays the name of the primary key used in the search and the range of key values in both characters and hexadecimals. In this example, the optimizer selected key EMPNO as the primary traversal key with the range of key character values zero to 00374 and hexadecimal values of 0000000000 to F0F0F3F7F4.

SEC. KEY

If there was a secondary key used in the search, its name would be displayed after the primary key and it would be followed by the range of key values in both characters and hexadecimals.

INDX WRK TBL SIZE (IN HEX)

The presort space (in hexadecimal) is 00003C70.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     5
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

*** CMD= SELFR DBID= 001   TABLE= PMF USER= $ID1 $ID.CBSTS005(001) 000102    DATE= 1062003 TIME= 105417
********** REQUEST SPECIFICATION *************
DATAVIEW ELEMENT: EMDTA OFFSET IN RECORD: 00000 LENGTH: 00080
WHERE:   LT Z N EMDTA 0000 0005 V 0005 00375 F0F0F3F7F5
ORDER-BY: D E EMDTA 0075 0005 P Y
KEY DEF: EMPNO ( 0001 )          TYPE: NATIVE   ( D0 ) ROWS: 00005333
   FIELDS= 00000.005   ;
   CARD  = 000000200   ;
   STATS: 006E 0015 005C 000C 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
KEY DEF: STZIP ( 0002 )          TYPE: KEYINC=Y ( 40 ) ROWS: 00005333
   FIELDS= 00068.002   ; 00070.005   ;
   CARD  = 000000060   ; 000000200
   STATS: 012C 0000 00CC 0000 00C8 0014 00C8 0014 00C8 0014 00C8 0014 00C8 0014
PRI. KEY EMPNO
       FROM=
    (IN HEX)= 0000000000
         TO= 00374
    (IN HEX)= F0F0F3F7F4
INDX WRK TBL SIZE (IN HEX)............... 00003C70
**************** OPTIMIZATION ******************
CBSOR REASONS: 1-5: NYYNN 6: I 7-9: NYN 10-17: NYNNNNNN 18-21: YNYN   ❸
TYPE OPTIMIZATION......: POPULATION INDEPENDENT
INDEX BUFFER REFERENCES: 00000000
**************** SELECTION *********************
TYPE SELECTION.........: *DIRECT*                    ❹
**************** ORDERING **********************
TEMPORARY INDEX REQUIRED DUE TO:
     O ORDER-BY FIELDS
     O SIGNED NUMERIC FIELDS                         ❺
ENTRIES INDEXED........: 00000200
LOGIO..................: 00000609
**************** RETRIEVAL *********************
RETRIEVAL FROM TEMP INDEX NOT USED DUE TO DATA NOT AVAIL.
DATA RECORDS ACCESSED DUE TO: RETRIEVAL
***END OF SELFR CMD FOR SET  00000007
CBS SET NUMBER  00000007  BEING RELEASED
TOTAL NBR INDEX ENTRIES SCANNED 00000400                  ❻
TOTAL NBR DATA AREA ROWS READ   00000400
TOTAL NBR ROWS ACCEPTED         00000200
***END-OF-SET***
```

❸

The OPTIMIZATION section displays the following information:

CBSOR REASONS:

See the *CA Datacom/DB Database and System Administration Guide* to interpret CBSOR values for 1-5, 6, 7-9, 10-17, and 18-21.

TYPE OPTIMIZATION:

Optimization is POPULATION INDEPENDENT, so key EMPNO is selected as the primary traversal key *without* statistical population counting.

INDEX BUFFER REFERENCES:

Index buffer references are zero.

❹

The SELECTION section displays the following information:

TYPE SELECTION:

Selection is DIRECT.

❺

The ORDERING section contains the following information:

TEMPORARY INDEX REQUIRED DUE TO:

The ordering criteria, because a signed numeric field is specified.

ENTRIES INDEXED:

200 rows meet the selection criteria.

LOGIO:

This requires 609 index and data buffer references to build the temporary index.

❻

The RETRIEVAL section displays the following information:

RETRIEVAL FROM TEMP INDEX NOT USED DUE TO DATA NOT AVAIL.

Data records must be re-accessed during retrieval from the temporary index because element EMDTA is not completely contained in the temporary index key.

DATA RECORDS ACCESSED DUE TO: RETRIEVAL

During record selection, data records were accessed because the sub-element of EMDTA (starting at byte 75 for a length of 5, needed to build a temporary index key) was not contained in the key EMPNO.

*****END OF SELFR CMD FOR SET 00000007**

SELFR processing has completed for the designated set.

CBS SET NUMBER 00000007 BEING RELEASED

An explicit (SELPR) or implicit request to release the set was processed.

TOTAL NBR INDEX ENTRIES SCANNED

Number of index entries read in order to satisfy the SELxx requests for set.

TOTAL NBR DATA AREA ROWS READ

Note that 400 data records are accessed since each row must be accessed once to build the temporary index, and once to be returned in the work area.

TOTAL NBR ROWS ACCEPTED

200 rows were returned to requestor.

*****END-OF-SET*****

No more diagnostics for this set.

Since different fields are used for selection and ordering (with selection not an "equal" condition), either a temporary index or index scanning is required. However, adding the order-by field as the last field of the traversal key would eliminate access to the data records while building the temporary index.

Statistical Population Counting Report

This section of the PXX Report appears only if the Compound Boolean Selection Optimizer performs statistical population counting. Statistical population counting is performed, if more than one candidate index exists after population independent analysis.

This sample report starts counting index candidates SQ157 and SQ158 at level 1 of the index.

```
*** STATISTICAL POPULATION COUNTING REPORT ***


 ❶    ❷    ❸     ❹      ❺               ❻       ❻

GRP  LVL  KNAME CNTKR-B  LOW EST  (TINDX)  HIGH EST  (TINDX)
0001 0001 SQ157 NOT ENOUGH ENTRIES AT THIS LEVEL
0001 0001 SQ158 NOT ENOUGH ENTRIES AT THIS LEVEL
0001 0000 SQ157 00000002 00000001                    NBR ROWS, BLK CHGS
0001 0000 SQ158 0000000A 00000001                    NBR ROWS, BLK CHGS
0001 0000 SQ157 00000002 00000002                     WINNER

HEURISTICS HEADER xxxxx HEURISTICS BUCKETS xxxxx   ❼
```

❶

GRP - "OR" group number. An "OR" group consists of predicates joined by AND. Statistical population counting is performed separately for each "OR" group.

❷

LVL - Index level counted. Statistical population counting begins with the top level of the index, which is level 1 in this case. If a winner cannot be determined, counting is repeated at the next lower level where a more accurate count can be made.

❸

KNAME - name of candidate index. Some indexes may have been eliminated in the population independent phase and not appear here. Also, candidate indexes may be eliminated at a level and not be recounted at lower levels.

❹

CNTKR-B - "count key range blocks." This is the number of pointers to lower level blocks counted in the index scan range. The first and last pointer have been eliminated because the portion of entries in these blocks within the scan range is unknown.

NOT ENOUGH ENTRIES AT THIS LEVEL indicates less than three pointers were counted. If more than one index has this message, counting proceeds to the next lower level to obtain a more accurate estimate.

At level 0, this is the actual number of rows counted.

❺

LOW EST/HIGH EST - low and high estimates. The variance of the statistics used for estimation are used to give a low and high estimate. A winning candidate index has a high estimate that is lower than all other candidate's low estimate.

The estimated cost is the sum of the number of index blocks, changes in data area blocks, and temporary index cost. Temporary index cost is computed as one per 10 rows indexed, plus one per 2 rows read from the temporary index, unless retrieval is from the temporary index.

The FST (FOR FIRST) value may limit the number of rows read from the temporary index, or from the permanent index for candidate indexes not requiring a temporary index. It does not limit the rows read from a permanent index while building a temporary index, unless the FOR ANY option is specified.

At level 0, since an exact count has been made there is no range. The number in the LOW EST column is the number of data block changes encountered during the count, except when WINNER or ELIMINATED appears on the right. Then, it is the total estimated cost.

❻

(TINDX) - estimated cost of building a temporary index. If some candidate indexes, but not all candidate indexes, require building a temporary index, this cost is included in the total estimated cost shown in the low/high estimate columns.

❼

HEURISTICS - When level 0 is counted, the key ID of the winning index is saved. Before counting level 0, a search is made for a heuristic record where the same index has been selected 8 out of the last 10 times that level 0 was counted. If found, that index is used and level 0 is not counted.

The HEADER data identifies the query. The BUCKETS data contains the last 10 winning key IDs.

Heuristic records are only kept for requests that uniquely identify a query in the User Information Block. This is done when the FOR statement is used. If you would like to use heuristics for call-level requests, uniquely identify each query in the first 29 bytes of the User Information Block, and place a valid packed decimal number in the last three bytes. For FOR statements, this number identifies the date of compilation because the query could have changed significantly in a new compilation.

Dumping SQL Internal Control Structures

You can dump SQL internal control structures held in memory. This feature should normally be used only at the request of Technical Support.

To dump the SQL internal control structures held in memory, generate an SQLCODE -999 (SQLSTATE 56S19) and the associated PXX or PXXSQL dumps using the following:

```
SELECT * FROM SYSADM.SQL_STATUS_DUMP;
```

For information about SQLCODE -999, see the *CA Datacom/DB Message Reference Guide*.

TYPE=DATANE - Native Sequence Efficiency Report

A DBUTLTY DATANE report provides a detailed native sequence efficiency report for the selected data table. The DATANE report provides detailed insight into the physical order of the data rows in comparison to the order of the Native (sequence) Key.

The information provided by the report is one of the key factors when considering whether a data table is a good candidate for reorganization. The CA Datacom/DB Data Reorganization Green Book provides a detailed best-practice process that includes these key information points:

- Native Efficiency Report (DATANE)
- Sequential accesses by the Native Key
- Sequential accesses by the non-Native Key
- Sequential accesses as a portion of the total table accesses.

The DATANE report uses a highly efficient index level process that has minimal effect on MUF processing. However, the DATANE function reads through the index for the table using one key and can do some limited data row retrievals. The DATANE report must be executed while the MUF is active and the database table is available at the least for read processing.

The DATANE report can process information while the table is open for update by other users. Update tasks in the MUF that add, delete, or move data rows affect the accuracy of the DATANE report. We recommend that the DATANE report is executed when there is a minimal amount of update activity against the selected data table. The DATANE report cannot be executed against a table that is locked for single-user processes such as a DBUTLTY LOAD.

The DATANE report builds a simple report that shows the detailed data row sequential efficiency for a selected data table. The report can help you determine whether to reorganize the table.

In most cases, run the DATANE using the Native Key. You can also use a key other than the Native Key. This ability allows you to compare the data row efficiency with various keys. In some cases, you want to see if a different key would better match the data row order. In other cases, you have several keys that are similar to the Native Key and you want to measure the efficiency of those keys.

Before the DATANE report, most users relied on various sources of information to determine whether to reorganize the table. The sources were the number of overflows in the data area, batch job run times, and so on. In most cases, this information was limited in its effectiveness.

Restrictions

The MUF must be active and operational and the data table must be available at least for read processing.

The table selected for the report must be a valid CA Datacom/DB data table. The table cannot be open by a single user function such as a DBUTLTY LOAD function.

The data area must be loaded with URI=YES.

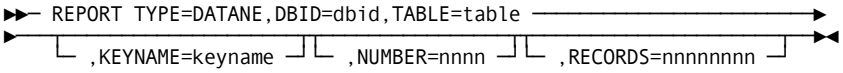
Security Setting: The DBUTLTY external resource name for the DATANE report is REPORT.DATANE.

When to Use

Use the DATANE report when you want to determine the sequential efficiency of the data table when compared to the Native (sequence) Key. The DATANE report also provides detailed information about the current data row sizes for compressed tables. The report also provides information about the rows and data blocks in use.

How to Use

To obtain a Native Efficiency report, execute the REPORT function with the following command format:



REPORT

Invokes the report function.

TYPE=

(*Required*) Requests the Native Efficiency Report.

Valid Entries:

DATANE

Default Value:

(No default)

,DBID=

(*Required*) Specifies the name of the DBID that you want to use.

Valid Entries:

a valid DBID

Default Value:

(No default)

,TABLE=

(*Required*) Identifies the table on which you want a report.

Valid Entries:

3-character DATACOM-NAME of the table in the database specified

Default Value:

(No default)

,KEYNAME=

(*Optional*) Specifies the key name to use to process the report. The data row sequential efficiency is compared to the order provided by this index.

If a value is not specified, the report uses the Native Key specified for the table.

Valid Entries:

5-character DATACOM-NAME of the key in the table specified

Default Value:

The Native Key of the selected table

,NUMBER=

(*Optional*) Specifies the number of data blocks that are read using physical processing (at the beginning of the report execution) to determine the maximum rows per block (MRB). The NUMBER parameter is only honored for data tables that are using compression. For non-compressed tables, the MRB is calculated using the row size and data block size.

Note: Specifying a low value limits the ability to calculate a valid MRB for this table. When the NUMBER parameter is not specified and the data table is compressed, the maximum value (9999 data blocks) is scanned to provide the MRB calculation.

Valid Entries:

0 to 9999

Default Value:

9999 (but only used if table is compressed)

,RECORDS=

(*Optional*) Specifies the rows to process (and include) in a printed group detail line.

If a value is not specified, no detail lines are produced and only the summary lines for the full table are produced.

Note: Specifying a low number for the RECORDS= parameter with a table with many rows generates many detail lines (one per record group). If the number of groups generated exceeds 99,999, the report allows the group number to roll over to zero and begin again. This rollover typically means that over 2000 pages of detail group lines are generated. We recommend that you specify a RECORDS= value that keeps the number of detail groups to 1000 groups or less.

Valid Entries:

0 to 99999999

Default Value:

(No default)

Example JCL: DATANE without RECORDS= Specified

The following example shows the command to obtain a DATANE summary report when the optional parameters KEYNAME, NUMBER, and RECORDS are not specified. These omissions tell the utility to produce a report using the Native Key and only produce the summary lines for the full table.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN    DD *                                  Command Input
        REPORT TYPE=DATANE,TABLE=REO,DBID=1022
/*
```

Sample Report: DATANE without RECORDS= Specified

Following are sample report pages. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  REPORT TYPE=DATANE,TABLE=REO,DBID=1022

                  FUNCTION=REPORT
                     DBID=01022
                     TABLE=REO
                     TYPE=DATANE
```

This page of the report shows the following:

- The command exactly as entered.
- An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.
- Any messages related to syntax processing.

```
TBL AREA KEY   CMP USER COMPRESSION   RECLN       RECORDS BLKSIZE     TRACKS   TOT BLKS  INUSE BLKS  PART BLKS
REO REO  REOK1 N                        112        31,950   4,096        150      1,800         890          1


CALC MAX ROWS/BLOCK              36
```

This page of the report shows the following information about the table that is being processed:

Top Heading and Lines

Displays the heading for the columns containing information describing the selected data table.

CALC MAX ROWS/BLOCK

Displays the maximum row per block (MRB) which was calculated by dividing the expanded row size plus RCE (record control element) into the data block size.

```
                              *--- BLOCK --*    PBC   887 MRB  *---- MAX ROW/BLK EFFICIENCY RATES WITH BUFFERS  ----* GROUP        ROWS
BLOCKS  MAX  MIN  AVG         BLOCK EFF   2   4   8  12  16  20  24  28  32  48  64  96 128 256
                READ     W/DATA ROWS ROWS ROWS       CHANGES  %   %   %   %   %   %   %   %   %   %   %   %   %   %   %   %


_____________________________________________________________________________________________________________________________________
FULL TABLE
MRB           31,950        888   36   18   36         6,931  13                                                        14  16
AMRB          31,950        888   36   18   36         6,931  13                                                        14  16
```

This page of the report shows the following efficiency ratings for the FULL TABLE:

PBC in the top heading

Displays the PERFECT BLOCK CHANGE (PBC) count of 887. The PBC is the number of block changes that would occur when a table was loaded with all rows in sequential order and all blocks had the maximum number of rows per block. The next lines are the headings for the detail line area of the report. Because RECORDS was not specified, no detail lines were generated.

FULL TABLE section

Displays the summary section of the report. Two lines are created for the table.

The **MRB** line provides the sequential efficiency rating of this table compared to an ideal or perfect situation where each block is loaded to the maximum (rows) and all the rows are in sequential order.

The **AMRB** (Adjusted MRB) line provides the sequential efficiency rating of this table using only the sequential order of the rows as the ideal. This rating does not penalize the efficiency rating for available free space that can be in the data blocks.

Note: Having large free space amounts could substantially reduce sequential performance by limiting the rows returned in a data block.

31,950

Indicates the number of rows in the table.

888

Indicates the number of data blocks that contained at least one data row for this table. The value varies slightly from the BLOCKS INUSE figure on page 2. The difference is data blocks that are used as data area control blocks.

36 18 36

Indicates the high-level mark (MAX. ROWS) and the low-level mark (MIN. ROWS) of each block having a row for this table. The report also divides the total rows of the table by the blocks in-use to calculate the average rows (AVG. ROWS).

6,931

Indicates the BLOCK CHANGES that occur when the data rows are processed using the selected key sequence. A BLOCK CHANGE is counted each time the next row (in the sequence given by the index chosen with the KEYNAME=) is not within the same block as the current row.

The BLOCK CHANGES is the key factor in measuring sequential efficiency. When a table is in sequential order, all data rows are processed in a given data block before moving on to the next block. When rows are not in order, the sequential process bounces back and forth between data blocks to retrieve all data rows in the selected key sequence. The bouncing back and forth adds overhead and delays to any process that is reading the data in sequential order.

13%

Indicates the MRB efficiency which is calculated by taking the ideal situation where all data blocks are only processed once as data rows are read sequentially (PBC 887) and dividing that by the actual number of block changes (6,931) that would have occurred. In other words, it is the efficiency of reading the data with a single buffer.

14 16

Displays the calculated MRB efficiency with buffers. Because buffering can significantly reduce the physical I/O used when a previously read data block is already in storage (a buffer), the MRB efficiency percentage is calculated using two buffers, four buffers, and so on.

A table that shows a high MRB efficiency percent (more than 80 percent) with buffers of 8 or less still provides a reasonable good sequential processing. There is processing overhead to find data blocks in the buffers, but it is significantly less than having to reread the data blocks from the DASD.

Some sites can have higher buffer availability for batch processing. For these sites, additional efficiency ratings are generated for 12 buffers, 16 buffers, and so on.

By monitoring the amount of sequential activity and the sequential efficiency of a table, you are able to make more informed decisions on when to do a data reorganization.

Note: The FULL TABLE or summary lines look at the table as a whole and their statistical calculations are based on the whole table.

Example JCL: DATANE with RECORDS= Specified

The following shows the command to obtain a DATANE detail report when the RECORDS= parameter contains a value. In this detail report, we are requesting a detailed efficiency line for every 1000 records. By generating detail lines, you are able to determine how the efficiency ratings vary within the table. The variations can occur when a data table has been reorganized into sequential order, but follow-up processing has deleted and added rows in a random fashion in one or more areas of the table.

The online reorganization process allows you to limit the data rows that are being reorganized (FIRSTKEY and LASTKEY). A DATANE detail report with a reasonable number of detail lines can assist you in determining if such a limited OLREORG is helpful.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN    DD *                                  Command Input
  REPORT TYPE=DATANE,TABLE=REO,DBID=1022,RECORDS=01000
/*
```

Sample Report: DATANE with RECORDS= Specified

Following are sample report pages. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    REPORT TYPE=DATANE,TABLE=REO,DBID=1022,RECORDS=01000

                    FUNCTION=REPORT
                       DBID=01022
                       RECORDS=00001000
                       TABLE=REO
                       TYPE=DATANE
```

This page of the report shows the following:

- The command exactly as entered.
- An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.
- Any messages related to syntax processing.

```
TBL AREA KEY   CMP USER COMPRESSION   RECLN       RECORDS BLKSIZE     TRACKS   TOT BLKS  INUSE BLKS  PART BLKS
REO REO  REOK1 N                       112        31,950   4,096        150      1,800         890          1


CALC MAX ROWS/BLOCK              36
```

This page of the report shows the following information about the table that is being processed:

Top Heading and Lines

Displays the heading for the columns containing information describing the selected data table.

CALC MAX ROWS/BLOCK

Displays the maximum row per block (MRB) which was calculated by dividing the expanded row size plus RCE (record control element) into the data block size.

The next and following pages of the detail report provide the efficiency ratings for each group of records. At the end of the report, the FULL TABLE summary lines are generated.

```
                        *--- BLOCK --*     PBC    27 MRB  *---- MAX ROW/BLK EFFICIENCY RATES WITH BUFFERS  ----*
 GROUP          ROWS     BLOCKS  MAX  MIN  AVG        BLOCK EFF   2   4   8  12  16  20  24  28  32  48  64  96 128 256
                READ     W/DATA ROWS ROWS ROWS       CHANGES  %   %   %   %   %   %   %   %   %   %   %   %   %   %


     1         1,000        122   17    1    8          122  23
     2         1,000        178   16    1    6          177  16
     3         1,000        266   17    1    4          272  10                                                      16  17
     4         1,000        256   19    1    4          261  11                                                  12  13  20
     5         1,000        293   14    1    3          292   9                                  10          11
     6         1,000        193   26    1    5          192  15                                                  16  19  42
     7         1,000        185   13    1    5          184  15                                                      17
     8         1,000        194   36    1    5          193  15
     9         1,000        138   36    1    7          137  21                                                      35
    10         1,000        241   15    1    4          263  11                                              12      16
    11         1,000        209   16    1    5          252  11                                                      16
    12         1,000        204   10    1    5          203  14                                                      18
    13         1,000        214   36    1    5          213  13
    14         1,000         47   36    4   21           46  63
    15         1,000        267   19    1    4          307   9                                              10      14
    16         1,000        221   15    1    5          243  11                      12                      13
    17         1,000        304   14    1    3          304   9                                                      13
    18         1,000        190   11    1    5          189  15
    19         1,000        299    8    1    3          299   9
    20         1,000        226   16    1    4          241  12
    21         1,000        165   15    1    6          174  16                                          18
    22         1,000        240   25    1    4          293   9              10                       11              13
    23         1,000        188   12    1    5          187  15                                                      16
    24         1,000        101   28    1   10          173  16          17                                      29  37
    25         1,000        243   20    1    4          245  11                                                      12
    26         1,000        232   17    1    4          240  12
    27         1,000        230   34    1    4          251  11                                                      14
    28         1,000        231   13    1    4          231  12
    29         1,000        137   24    1    7          178  16                                                  18  24
    30         1,000        154   16    1    6          159  18                                          19          100
    31         1,000        282   13    1    4          288  10                                                      11
    32           950        122   32    1    8          122  22
_____________________________________________________________________________________________________________________
FULL TABLE
MRB           31,950        888   36   18   36        6,931  13                                                  14  16
AMRB          31,950        888   36   18   36        6,931  13                                                  14  16
```

PBC 27

Displays the PERFECT BLOCK CHANGE (PBC) count for the selected group of records (1000). The PBC is the number of block changes that would occur if a table was loaded with a group (1000) of rows in sequential order and all blocks had the maximum number of rows per block.

The following lines display the detail line area of the report. Because RECORDS=1000 is specified, there is one line of output for each group of 1000 rows.

GROUP

Displays the group number being processed. The group number begins at 1 and increments until it reaches 99,999 groups. If this number of groups is reached, the report continues with the next group number being with zero.

Note: Each group is processed autonomously from the other rows in the table. The goal is to establish efficiency for this group of rows. When a data block has 30 rows but only 15 rows are from this group, only those 15 rows are recognized and the rest of the rows in the block are ignored. This fact must be remembered when you are evaluating the statistical values in the columns. You cannot assume that the detail group line values add up to match the FULL TABLE summary values.

ROWS READ

Displays the number of rows in the group.

BLOCKS W/DATA

Displays the number of data blocks that contained at least one data row for this group.

BLOCK: MAX ROWS, MIN ROWS and AVG ROWS

Displays an evaluation for each group. Identifies the high-level mark (MAX ROWS) and the low-level mark (MIN ROWS). The report also divides the total rows of the group by the blocks with data rows for the group to calculate the average rows (AVG. ROWS).

BLOCK CHANGES

Displays the number of changes that would occur when the groups data rows were processed using the selected key sequence (chosen using KEYNAME=). A BLOCK CHANGE is counted each time the next row in the sequence is not within the same block as the current row.

The BLOCK CHANGES is the key factor in measuring sequential efficiency. When a table is in sequential order, all data rows are processed in a given data block before moving on to the next block. When rows are not in order, the sequential process bounces back and forth between data blocks to retrieve all data rows in the selected key sequence. This bouncing back and forth adds overhead and delays to any process that is reading the data in sequential order.

EFF %

Indicates the MRB efficiency which is calculated by taking the ideal situation where all data blocks are only processed once as data rows are read sequentially in the group (PBC 27) and dividing that by the actual number of block changes that actually occurred.

MAX ROW/BLK EFFICIENCY RATES WITH BUFFERS

Displays the calculated MRB efficiency with buffers. Because buffering can significantly reduce the physical I/O used to read a previously read data block back into storage, the MRB efficiency percentage is recalculated using two buffers, four buffers, and so on.

FULL TABLE section

Summarizes the report. Two lines are created for the table. This information was described earlier in this section.

Note: The detail group lines treat each group autonomously and generate their statistical calculations based on that group. The total or summary lines look at the table as a whole and their statistical calculations are based on the whole table. You cannot assume that the detail group line values add up to match the FULL TABLE summary values.

Example JCL: DATANE for a Compressed Table with NUMBER=0200 Specified

he following example shows the command to obtain a DATANE detail report when the table is compressed and the NUMBER= parameter contains a value. For this report, we specified to use the first 200 physical blocks (NUMBER=0200) to determine the MRB value.

We also specified a detailed efficiency line for every 1000 records.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN    DD *                                  Command Input
  REPORT TYPE=DATANE,TABLE=REO,DBID=1023,RECORDS=01000,NUMBER=0200
/*
```

Sample Report: DATANE for a Compressed Table with NUMBER=0200 Specified

Following are sample report pages. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    REPORT TYPE=DATANE,TABLE=REO,DBID=1023,RECORDS=01000,
                           NUMBER=0200

                    FUNCTION=REPORT
                       DBID=01023
                       NUMBER=0200
                       RECORDS=00001000
                       TABLE=REO
                       TYPE=DATANE
```

This page of the report shows the NUMBER=0200 parameter has been specified to limit the physical blocks scanned to determine the MRB value. The scan is only done when a table has compression. This page also shows the following:

- The command exactly as entered.
- An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.
- Any messages related to syntax processing.

The second page of the report is similar to what you have seen earlier in this section. However, note the following specific differences.

```
TBL AREA KEY   CMP USER COMPRESSION   RECLN       RECORDS BLKSIZE     TRACKS   TOT BLKS  INUSE BLKS  PART BLKS
REO REO  REOK1 Y                       100        31,950   4,096        150      1,800           2          0


SCAN MAX ROWS/BLOCK             108     #ROWS READ        18,260
```

CMP Y

Indicates that this table is compressed. Because the USER COMPRESSION field is blank, we know that this table is compressed using CA Datacom/DB standard compression.

SCAN MAX ROWS/BLOCKS

Indicates that a physical scan of the data blocks was used to determine the MRB value. The number of blocks scanned was specified in the NUMBER=0200 parameter. In scanning the first 200 data blocks, the MRB value was determined to be 108.

#ROWS READ

Indicates that, in scanning the first the 200 data blocks, the report encountered 18,260 data rows.

The third and following pages of the detail report provide the efficiency ratings for each group of records. At the end of the report, the FULL TABLE summary lines are generated.

```
                          *--- BLOCK --*     PBC     9 MRB  *---- MAX ROW/BLK EFFICIENCY RATES WITH BUFFERS  ----*
 GROUP          ROWS     BLOCKS  MAX  MIN  AVG        BLOCK EFF   2   4   8  12  16  20  24  28  32  48  64  96 128 256
                READ     W/DATA ROWS ROWS ROWS       CHANGES  %   %   %   %   %   %   %   %   %   %   %   %   %   %

     1         1,000        113   38    2    9          113   7
     2         1,000        152   18    1    7          151   5
     3         1,000         93   30    2   11          103   8                                                 9                  81
     4         1,000         85   34    1   12          106   8                                                    10              14
     5         1,000         62   48    1   16           61  14
     6         1,000         81   74    2   12           80  11                                                                15  56
     7         1,000        155   21    1    6          154   5                                                                    10
     8         1,000        116   25    1    9          123   7                                                                    15
     9         1,000         90   39    1   11          110   8                                                       9  10       25
    10         1,000         56   43    5   18           56  16
    11         1,000         66   26    1   15           65  13                                                                21
    12         1,000        139   22    1    7          138   6                                                                    15
    13         1,000        170   22    1    6          169   5
    14         1,000         99   42    1   10          100   9                                                             10  12  50
    15         1,000         54   60    1   19           53  16                                                                18  19
    16         1,000         59   26    1   17           58  15                                                                16
    17         1,000        111   32    1    9          111   8                                                                    56
    18         1,000        138   40    1    7          137   6                                                                     7
    19         1,000         86   43    2   12           93   9                                                    10              100
    20         1,000         86   33    1   12          113   7                                                 8   9  11          14
    21         1,000         66   49    1   15           65  13
    22         1,000        108   27    1    9          107   8                                                                    50
    23         1,000        126   18    1    8          125   7                                                                     8
    24         1,000        110   54    1    9          109   8                                                                    24
    25         1,000         93   57    1   11           94   9                                                          11  14       25
    26         1,000         79   31    1   13           78  11                                                    14
    27         1,000         91   24    1   11           90  10                                                                    20
    28         1,000        119   21    1    8          119   7                                                                    11
    29         1,000        124   36    1    8          123   7                                                                    11
    30         1,000         90   33    1   11           90  10                                                          11  13       69
    31         1,000         94   31    1   11           93   9                                                    10              12
    32           950         56   27    5   17           55  16
-------------------------------------------------------------------------------------------------------------------------------------
FULL TABLE
MRB           31,950        387  108   36   83        3,242   9                                                                10  15
AMRB          31,950        387  108   36   83        3,242  11                                                12              13  20
```

PBC 9

Displays the PERFECT BLOCK CHANGE (PBC) count for the selected group of records (1000). The PBC is the number of block changes that would occur if a table was loaded with a group (1000) of rows in sequential order and all blocks had the maximum number of rows per block. A compressed table allows more rows to fit per block as evidenced by the increased MRB (from 36 to 108), which in turn lowers the PBC value.

The rest of the entries on the report were described in the previous samples.

TYPE=DATASP - Data Area Space Report

The Data Area Space (DATASP) Report provides a detailed space usage map for a selected data area. This function processes the data area using a high speed I/O process analyzing each data block in the data area. The DATASP report provides detailed insight into the data block space usage and available free space.

The DATASP report uses a highly efficient I/O process that has no effect on the MUF processing. The DATASP function reads the data area in place and does not see activity currently occurring in the MUF that is not yet posted to the data area. The DATASP report can be executed when there is only a minimal amount of update activity against the selected data area.

The purpose of the DATASP report is to build a simple report that shows the detailed space usage for a selected data area that can help you determine, among other things, the amount of rows that could be added in the space available in the data area.

Before the DATASP report, most users relied on the information in the CXX report or Dynamic System Tables to determine the “fullness” of a data area. This information is provided in the following CXX report columns:

BLOCKS IN USE - 87

BLOCKS UNUSED - 12

TOTAL BLOCKS - 99

PERCENT FULL - 88

While this information provides a reasonably accurate accounting of the data blocks usage, it may not accurately reflect the data row “fullness.” For example, if a table was loaded with 900 data rows and those rows filled 90 of 100 available data blocks, both the data block and data row “fullness” in the data area would be approximately 90% full. If following the load, an application process deleted 90 data rows where each data row just happened to be in the next data block, the data block fullness would still be at 90%, but the data row fullness would only be 81%.

In many cases, the difference between data block fullness and data row fullness will be minimal. A few examples of where this would occur are the following:

- Data areas with DSOP 0
- Data areas with DSOP 1 where row sizes in the data area are similar and more rows are added than are deleted
- DSOP 3 without a cluster key where row sizes in the data area are similar and more rows are added than are deleted.

In certain other cases, the difference between data block fullness and data row fullness may be significant. A few examples of where this could occur are the following:

- Data areas with DSOP 1 where more rows are deleted than are added
- Data areas with DSOP 2
- Data areas with DSOP 3 where a cluster key has been selected
- Data areas with DSOP 4 after an OLREORG function
- Data areas with DSOP 5

The DATASP report gives you a simple and accurate way to access the data row fullness for a data area. The report also provides other information about the data rows that is beneficial for you to know. For example, it can provide the maximum, minimum, and average compressed row size in a compressed table.

Restrictions

The area selected for the report must be a valid CA Datacom/DB data area. The area cannot be open by a DBUTLTY INIT or DBUTLTY LOAD function. The data area must be loaded with URI=YES.

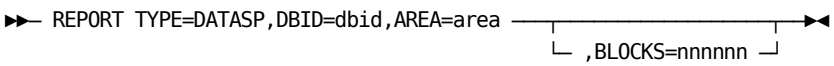
Security Considerations: The DBUTLTY external resource name for the DATASP report is REPORT.DATASP.

When to Use

Use the DATASP report when you want to determine how full the current data area is and the available row space in a data area. The DATASP report also provides detailed information about the available free space, the number and size of data rows, and various other data row information.

How to Use

To obtain a Space Usage (DATASP) report, execute the REPORT function with the following command format:



REPORT

Invokes the report function.

TYPE=DATASP

(*Required*) Requests the data area space usage report.

Valid Entries: DATASP

Default Value: (No default)

,DBID=

(*Required*) Specifies the number of the DBID that you want to use.

Valid Entries: A valid DBID

Default Value: (No default)

,AREA=

(*Required*) Identifies the area you want to report on.

Valid Entries: DATACOM-NAME of the area in the database specified

Default Value: (No default)

,BLOCKS=

(*Optional*) Specifies the blocks to process (and include) in each printed detail line output. If a value is not specified, no detail lines are produced and only the summary lines for the data area are produced.

Note: Specifying a low number for BLOCKS= parameter with a data area with a large amount of data blocks generates a large number of detail lines (one per block group). If the number of groups generated exceeds 99,999 groups, the report allows the group number to “roll” over to zero and begin again. This typically would also mean that over 2000 pages of detail group lines would be generated. We recommend that a BLOCKS= value be specified that keeps the number of detail groups to a 1000 groups or less.

Valid Entries: 0 to 999999

Default Value: (No default)

Example JCL: DATASP without BLOCKS= Specified

The following shows the command to obtain a DATASP report when BLOCKS= is not specified.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
//jobname     See the previous note and JCL Requirements.
//        EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB     See the previous note and JCL Requirements.
//CXX       DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN     DD *                                  Command Input
          REPORT TYPE=DATASP,DBID=1,AREA=PAY
/*
```

Sample Report: DATASP without BLOCKS= Specified

Following are sample report pages. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  REPORT TYPE=DATASP,DBID=1,AREA=PAY

                  FUNCTION=REPORT
                     AREA=PAY
                     DBID=00001
                     TYPE=DATASP
```

This page of the report shows the following:

- The command exactly as entered.
- An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.
- Any messages related to syntax processing.

```
AREA PAY  BLKSIZE  1,024  TRACKS         2  BLOCKS        66  URI YES  DSOP 0 NO RECLAMATION


TBL  CMP  USER COMPRESSION   RECLN       RECORDS
PAY                            52           200

 GROUP#      BLOCKS       KBYTES            ROWS FREESP  ROWS ROWLEN   ---------- FREESPACE IN BLOCKS ----------
             IN-USE  %    IN-USE   %       IN-USE    MAX   MAX    MAX   0 TO 1/4K       1/2K         1K          2K
             EMPTY   %     EMPTY   %      DELETED    MIN   MIN    MIN          3K         4K          8K         12K
                                           *AVAIL    AVG   AVG    AVG         16K        20K         24K         32K

TOTALS         13  17        10  16          200  1,022    19     52          10          1         53          0
               53  83        54  84            0     34     0     52           0          0          0          0
                                           1,016    860     3     52           0          0          0          0
```

This page of the report shows the following:

AREA

Displays the area DATACOM-NAME, PAY.

BLKSIZE

Displays the blocksize in the area, 1,024.

TRACKS

Displays the number of tracks used, 2.

BLOCKS

Displays the number of blocks, 66.

URI

Displays whether URI is assigned, YES.

DSOP

Identifies that the data space option is 0 with no reclamation.

The next lines are created for each table in the area.

TBL

Displays the table DATACOM-NAME, PAY.

CMP

Displays whether this is a compressed table.

USER COMPRESSION

Contains the user exit name when this table has a user compression routine specified or blanks when CA Datacom/DB standard compression is used.

RECLN

Displays record length including the RCE bytes, 52.

RECORDS

Displays the number of records, 200.

DSOP

Identifies that the data space option is 0 with no reclamation.

BLOCKS IN-USE % EMPTY %

Displays the number of blocks in use followed by the percentage in use over the number of blocks empty followed by the percentage of blocks empty.

KBYTES IN-USE % EMPTY %

Displays the number of kilobytes in use followed by the percentage in use over the number of kilobytes empty followed by the percentage of kilobytes empty.

ROWS IN-USE DELETED *AVAIL

Displays the number of rows in use, rows that are currently marked logically deleted, and number of rows could be added. Available rows are a best estimate based on average row size, space available in data blocks, and so on.

FREESP MAX MIN AVG

Displays the size of the largest segment of free-space bytes (MAX), the smallest segment of free-space bytes (MIN), and the average segment size of free-space bytes (AVG).

ROWS MAX MIN AVG

Displays the largest number of rows found in any data block (MAX), smallest number of rows found in a data block (MIN), and average number of rows per data block (AVG).

ROWLEN MAX MIN AVG

Displays the largest row length found in any data block (MAX), smallest row length found in a data block (MIN), and average row length per data block (AVG).

FREESPACE IN BLOCKS

Displays a map of free-space segments in each size grouping.

Example JCL: DATASP with BLOCKS= Specified

The following shows the command to obtain a detail DATASP report when “BLOCKS=10” is specified.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN    DD *                                  Command Input
         REPORT TYPE=DATASP,DBID=1,AREA=PAY,BLOCKS=10
/*
```

Sample Report: DATASP with BLOCKS= Specified

Following are sample report pages. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  REPORT TYPE=DATASP,DBID=1,AREA=PAY,BLOCKS=10

                  FUNCTION=REPORT
                     AREA=PAY
                     BLOCKS=000010
                     DBID=00001
                     TYPE=DATASP
```

This page of the report shows the following:

- The command exactly as entered.
- An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.
- Any messages related to syntax processing.

```
AREA PAY  BLKSIZE  1,024  TRACKS         2  BLOCKS        66  URI YES  DSOP 0 NO RECLAMATION


TBL  CMP  USER COMPRESSION   RECLN       RECORDS
PAY                            52           200

 GROUP#      BLOCKS       KBYTES            ROWS FREESP  ROWS ROWLEN   ---------- FREESPACE IN BLOCKS ----------
             IN-USE  %    IN-USE   %       IN-USE    MAX   MAX    MAX   0 TO 1/4K       1/2K         1K          2K
             EMPTY   %    EMPTY    %     DELETED    MIN   MIN    MIN          3K         4K          8K         12K
                                         *AVAIL    AVG   AVG    AVG         16K        20K         24K         32K

      1        10 100       10  97          190     34    19     52          10          0           0          0
               0   0        0   3            0     34    19     52           0          0           0          0
                                             0     34    19     52           0          0           0          0



      2         1  10        1   5           10  1,022    10     52           0          1           9          0
                9  90        9  95            0    502     0     52           0          0           0          0
                                           180    970     1     52           0          0           0          0

      3         0   0        0   0            0  1,022     0      0           0          0          10          0
               10 100       10 100            0  1,022     0      0           0          0           0          0
                                           190  1,022     0      0           0          0           0          0

      . . . . . .

      6         0   0        0   0            0  1,022     0      0           0          0          10          0
               10 100       10 100            0  1,022     0      0           0          0           0          0
                                           190  1,022     0      0           0          0           0          0

      7         0   0        0   0            0  1,022     0      0           0          0           4          0
                4 100        4 100            0  1,022     0      0           0          0           0          0
                                            76  1,022     0      0           0          0           0          0

 TOTALS        13  17       10  16          200  1,022    19     52          10          1          53          0
               53  83       54  84            0     34     0     52           0          0           0          0
                                         1,016    860     3     52           0          0           0          0
```

This page of the report shows the following information describing the selected data area followed by information describing the table or tables in the area. One line is created for each table in the area.

Note: The columns under the following headings are filled in for each reference group as specified with the REFGROUP parameter on the DBUTLTY OLREORG function.

BLOCKS IN-USE % EMPTY %

Displays the number of blocks in use followed by the percentage in use over the number of blocks empty followed by the percentage of blocks empty.

KBYTES IN-USE % EMPTY %

Displays the number of kilobytes in use followed by the percentage in use over the number of kilobytes empty followed by the percentage of kilobytes empty.

ROWS IN-USE DELETED *AVAIL

Displays the number of rows in use, rows that are currently marked logically deleted, and number of rows could be added. Available rows are a best estimate based on average row size, space available in data blocks, and so on.

FREESP MAX MIN AVG

Displays the size of the largest segment of free-space bytes (MAX), the smallest segment of free-space bytes (MIN), and the average segment size of free-space bytes (AVG).

ROWS MAX MIN AVG

Displays the largest number of rows found in any data block (MAX), the smallest number of rows found in a data block (MIN), and the average number of rows per data block (AVG).

ROWLEN MAX MIN AVG

Displays the largest row length found in any data block (MAX), smallest row length found in a data block (MIN), and average row length per data block (AVG).

FREESPACE IN BLOCKS

Displays a map of free space segments in each size grouping.

Note: Additional groups of blocks are listed with their information.

TOTALS

Displays a summary of all group information.

Recovery File (RXX) Report

This section contains information about:

- Recovery File (RXX) Analysis Report (see following section)
- Split File Analysis Report (see Split File Analysis Report (see page 642))

Recovery File (RXX) Analysis Report

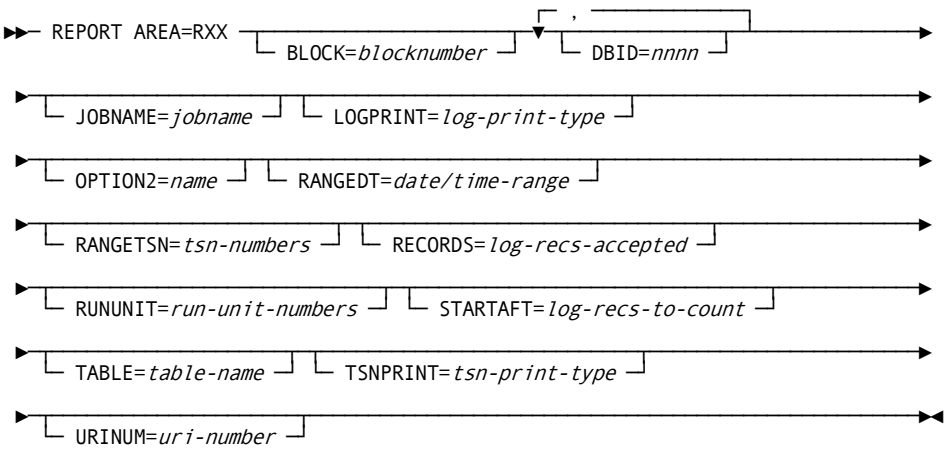
A Recovery File (RXX) Analysis Report reads a RXX and produces a summary report of the contents.

The many options provided by this report allow you to use information about the contents of a Recovery File to analyze logging and recovery operations and correct any problems that could occur.

This utility always reads one input file of one or more volumes. In all environments, if the input is not log data, unexpected results occur.

Following is the syntax for the RXX Analysis Report:

Note: The Recovery File Analysis Report program can detect the type of input file. If the input is a CA Datacom Fast Restore file (split), the Report control statements are as shown in the Split File Analysis Report syntax on page Split File Analysis Report (see page 642).



The syntax just shown defines the provided and optional reports. Each was described as though the entire input was considered selected data (the default), but be aware that numerous data selection options are available. Remember, too, that a log record, to be selected, must pass all of the specified options (a single option can be specified or any combination, including all of them).

Each of the following options can only be specified once, except for the DBID= option, that you can repeat.

Command

REPORT

Invokes the function to produce a report.

Required Keyword

AREA=RXX

Requests the Recovery File Analysis Report.

Optional Keywords

BLOCK=

This is specified as an eight-character hexadecimal number representing a data block within a data area. This option would normally be included when requested by Technical Support.

DBID=

Use to specify a database ID. You can repeat this option. Each occurrence may specify one or more valid database IDs or a range of DBIDs nnnn-mmmm (a range, for example, such as 100-250).

JOBNAME=

Specify a job name.

LOGPRINT=

Choose either DETAIL or DETAILX for *log-print-type.*

The DETAIL option prints an INPUT RECORD that can be useful, if one execution selects all records for a transaction. This information can be used to dump the records around the selected records in another execution. If the same input file(s) are provided, this input record number is the same for each execution.

The DETAILX option prints an INPUT RECORD and formats a hex dump of the log record. As either of these options may produce large quantities, the standard DBUTLTY report headings are reduced to an abbreviated one line.

OPTION2=

Specify a data area name or the Full Parent name of a partitioned table.

RANGEDT=

Specify two date/times to provide a range (low to high). The date/times are specified as a 28 character number with the lower date/time immediately followed by the higher date/time. Each date/time is 14 characters in the format 'ccyymmddhhmmss'. For example: '20021214000000020020228235959'. Note that the input is not read beyond one record higher than this ending value.

RANGETSN=

Specify two TSN numbers to provide a range (low to high). The TSN numbers are specified as a 16-character hexadecimal number comprising 8 characters for the low TSN followed by the high TSN.

RECORDS=

Provides a count of log records that are *acceptable* after processing the number of log records specified in the STARTAFT= parameter. If STARTAFT= is not specified, the count is presumed to be zero. The combination of RECORDS= and STARTAFT= provides the ability to print/dump portions of a Recovery File (RXX). Valid entries for RECORDS= is a number from 0 through 99999999. The default is *all remaining records.*

Note: The input is not read beyond the number of records that are to be selected.

RUNUNIT=

Specify a run unit number to restrict record selection. Valid entries for RUNUNIT= is a number from 0 through 99999. The default is *all run units.*

STARTAFT=

Specify a count of log records before which all are rejected. Valid entries for STARTAFT= is a number from 0 through 99999999. The default is 0.

TABLE=

Specify a table name. This is the table name in the Request Area when the log record was built.

TSNPRINT=

Specify a transaction report that prints one line of output for every transaction selected in the input.

Choose either SUMMARY, DETAIL, or DETAILX for *tsn-print-type.* DETAIL and DETAILX are described in the LOGPRINT= section on page Recovery File (RXX) Analysis Report. Specifying SUMMARY produces the basic transaction report. For more information and examples see TSNPRINT=DETAIL Sample (see page 638).

URINUM=

Provides an eight-character URI number.

Example JCL

The following shows the command to generate a Recovery File Analysis Report.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYS005,cuu
 // TLBL RXX,'rxx'                                  RXX data set
 // EXEC DBUTLTY,SIZE=750K
       REPORT AREA=RXX,LOGPRINT=DETAIL,TSNPRINT=DETAIL,DBID=015,TABLE=MSG
 /*
 /&
 * $$ EOJ
```

Report AREA=RXX (DISK)

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
// ASSGN SYS001,DISK,VOL=volser,SHR
// DLBL RXX,'rxx'                                  RXX data set
// EXTENT SYS001,volser
// EXEC DBUTLTY,SIZE=750K
       REPORT AREA=RXX,LOGPRINT=DETAILX
/*
/&
* $$ EOJ
```

Sample Report: REPORT AREA=RXX

Following is a sample RXX Analysis Report for a database with an ID of 15 and a table named MSG, using LOGPRINT=DETAIL and TSNPRINT=DETAIL.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶           REPORT AREA=RXX,LOGPRINT=DETAIL,TSNPRINT=DETAIL,TABLE=MSG,DBID=15

                    FUNCTION=REPORT
        ❷             AREA=RXX
                      DBID=00015
                      LOGPRINT=DETAIL
                      TABLE=MSG
                      TSNPRINT=DETAIL
        ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

LOGPRINT=DETAIL Sample

Specifying LOGPRINT=DETAIL in an RXX Analysis Report generates the following information, which continues for as many pages as necessary (not all pages are shown in this sample).

```
                    (CONTINUED)                                  Date: mm/dd/ccyy   Time: hh.mm.ss    Page:      2
                                                LOGPRINT DETAIL REPORT
CMD   TBL DBID URI      BLOCK    L# TSN      JOB NAME R-UNIT MP USER ID AREA MONITOR INFO     INPUT RECORD FPR    RXX  }❶
      DATE       TIME     CLOCK            SEC.USER MOVE TO  LRID     PREV     LRSN     LBSN     TASK STATUS RECL DATA }

ADDIT MSG   15 00000401 00000003 00 00001A18 DLV1C131 17,003    SQLDG001 MSG SCFDG001JVM00037           38
      mm/dd/ccyy hh.mm.ss B6F7A1DA0D189903 username 00000000 00000608 00000607 00000008 00000006    1 B      0175 6C3B
ADDIT MSG   15 00000402 00000003 00 00001A18 DLV1C131 17,003    SQLDG001 MSG SCFDG001JVM00037           39
      mm/dd/ccyy hh.mm.ss B6F7A1DA0D1D7363 username 00000000 00000609 00000608 00000009 00000006    1 B      0175 6C3B
ADDIT MSG   15 00000403 00000003 00 00001A18 DLV1C131 17,003    SQLDG001 MSG SCFDG001JVM00037           40
      mm/dd/ccyy hh.mm.ss B6F7A1DA0D2152A3 username 00000000 0000060A 00000609 0000000A 00000006    1 B      0175 6C3B
         .
         .
         .
DELET MSG   15 00000401 00000003 00 00001A18 DLV1C131 17,003    SQLDG001 MSG SCFDG001JVM00037           83
      mm/dd/ccyy hh.mm.ss B6F7A1DB93E43B61 username 00000000 00000635 00000634 00000035 00000006    1 B      015E 6C3B
DELET MSG   15 00000402 00000003 00 00001A18 DLV1C131 17,003    SQLDG001 MSG SCFDG001JVM00037           84
      mm/dd/ccyy hh.mm.ss B6F7A1DB93E7C541 username 00000000 00000636 00000635 00000036 00000006    1 B      015E 6C3B
         .
         .
         .
```

❶

Log Detail Report Headings

CMD

CA Datacom/DB record-at-a-time command. In addition to record-at-a-time commands such as ADDIT, UPDAT, DELET, and COMIT, commands used internally can also appear in the report. Internal commands that support restart and recovery requirements are not normally documented, but because they might be seen in this report, a summary description follows.

- Commands that start with the letter Y, for example YYCTC, and DELAL are used to support CA Datacom/DB or SQL.
- Commands that start with LOGP relate to two-phase commit.
- Commands that start with LOG (not LOGP) exist when needed for request or transaction control.
- The CYCLE command is used to note the start and end of a spill.
- The MOVER command relates to the movement of records, both during compressed record maintenance and execution of the DBUTLTY OLREORG function. For more information about the MOVER command, see Request Summary Report.
- Other commands that exist and could also appear in the report are not under the control of users.
- We reserve the right to add new internal commands and modify or delete existing commands as needed.

TBL

CA Datacom/DB table name in the Request Area. Blank for records without table names. For 'CYCLE' (the internal command), the table name contains 'STA' (for start of a cycle) and 'END' (for end of a cycle). A cycle refers to a sequential group of log blocks copied from the LXX to the RXX during one spill.

DBID

Database ID. Blank for records without database ID's. Printed as a decimal number.

URI

Unique Row Indentifier. Blank for records without a URI number. Printed in hexadecimal format. Contains the literal 'NON-URI ' for areas not loaded as URI.

BLOCK

Data area block number. Blank for records without a data block. Printed in hexadecimal format.

L#

Data area block record number. Blank for records without a data block number. Contains '00' for records with a data block, if the area is loaded with the URI option. Printed in hexadecimal format.

TSN

Transaction Sequence Number. Blank for records not part of a transaction. Printed in hexadecimal format.

JOB NAME

Job name.

R-UNIT

Run unit number.

MP

Blank if not running r10 or later as a MUFplex. Otherwise contains a MUFplex number of which Multi-User Facility logged this log record.

USER ID

User ID. First 8 characters of the 32-byte user ID passed on the request.

AREA

Data area name containing row.

MONITOR INFO

Monitor information, if this request was made through the CA Datacom CICS Services or the CA Datacom Server.

INPUT RECORD

Number of this input log record.

FPR

Full Parent name, if this table is partitioned, or a partition.

RXX

To indicate this is a recovery file. In z/OS, can contain LXX.

DATE

Date the log record was added. Printed in the form mm/dd/ccyy.

TIME

Time the log record was added. Printed in the form HH.MM.SS.

CLOCK

Result of a hardware STCK (store clock) instruction at the time the log record was added.

SEC.USER

Security user requesting maintenance.

MOVE TO

Block number to where an update moved a record. Printed in hexadecimal format.

LRID

Log record ID. The physical block number within the LXX where the log record was added in the first three bytes. The last byte is a sequential number within this log block, such that the first record is 1, and so forth. This field is provided to assist Technical Support, should the need arise. Printed in hexadecimal format.

PREV

Previous log record ID. Log records for a transaction are chained in reverse back to a previous one of low values.

LRSN

Log record sequence number. An internal number used by CA Datacom/DB in support of the logging features. Incremented by one as log records are added, can restart at 1, based upon internal DB requirements. This field is provided to assist Technical Support, should the need arise. The number is printed in hexadecimal.

LBSN

Log block sequence number. An internal number used by Technical Support of the logging features. Incremented by one as log blocks are used. Can start at many numbers and can restart as required internally by CA Datacom/DB. This field is provided to assist Technical Support, should the need arise. The number is printed in hexadecimal.

TASK

Number of the task area which was used when this log record was generated.

STATUS

Status of log record. Can contain one or more letters. A blank ends the status. The possible letters are:

A Specifies that the transaction was aborted and is to be rolled back.

B Specifies that this record is subject to transaction backout.

F Specifies that the first log record after transaction was force spilled.

M Specifies an update command caused the record to move.

U Specifies that this request was rolled back.

V Specifies a virtual table.

1 Specifies the first logged record for this transaction.

X Specifies this log record is part of an ADDIT/DELET set, done in support of a partitioned table update that changed the partition in which the row is stored.

These letters may be in any order. They are present only when required/desired to support the CA Datacom/DB facilities. They are provided for Technical Support and are not always present even though the description would indicate that it should be present.

RECL

Size of the log record. It is printed in hexadecimal. This field is provided to assist Technical Support, should the need arise.

DATA

Size of the log data in this log block. It is printed in hexadecimal. This field is provided to assist Technical Support, should the need arise.

TSNPRINT=DETAIL Sample

Using the TSNPRINT=DETAIL option in an RXX Analysis Report generates the TSN DETAIL REPORT as shown in the following sample.

```
                    (CONTINUED)                                  Date: mm/dd/ccyy   Time: hh.mm.ss    Page:     43

                                                TSN DETAIL REPORT
TSN      JOB      FIRST RECORD        LAST RECORD            REQUESTS STAT TBL DBID        ADD      DELETE      UPDATE  ❺

00001A18 DLV1C131 mm/dd/ccyy hh.mm.ss mm/dd/ccyy hh.mm.ss          90 NN   MSG   15         45          45           0
00001A19 DLV1C131 mm/dd/ccyy hh.mm.ss mm/dd/ccyy hh.mm.ss          90 NN   MSG   15         45          45           0
00001A1A DLV1C131 mm/dd/ccyy hh.mm.ss mm/dd/ccyy hh.mm.ss          90 NN   MSG   15         45          45           0
00002252 DLV1C131 mm/dd/ccyy hh.mm.ss mm/dd/ccyy hh.mm.ss         124 NN   MSG   15         62          62           0
00002253 DLV1C131 mm/dd/ccyy hh.mm.ss mm/dd/ccyy hh.mm.ss         322 NN   MSG   15        161         161           0
00002280 DLV1C131 mm/dd/ccyy hh.mm.ss mm/dd/ccyy hh.mm.ss         304 NN   MSG   15        152         152           0
```

❺

TSN Detail Report Headings

TSN

The transaction sequence number.

JOB

The name of the job.

FIRST RECORD

The date and time the first record in a transaction was processed.

LAST RECORD

The date and time the last record in a transaction was processed.

REQUESTS

The total number of requests for the transaction, that is, the sum of the number of ADD, DELETE, and UPDATE requests.

STAT

The first character under the STAT heading is a 1 (a numeral one) if the first record of the transaction was found among the selected input, or the first character is an N (NO) if the first record was not found. The second character in the STAT field is a Y (YES) if the transaction completion record was found among the selected input, else an N.

An indication that a transaction completed does not necessarily mean that it completed *successfully*. Some or all of the counts of adds, deletes, and updates may have been reversed and thereby nullified. The reversal can be done *inline* to the original log record or it can be done with an additional reversal log record. For example, an add could be reversed with a delete. Deletes are usually reversed as an update, but an add is possible. If these are concerns, then the detail log records would have to be reviewed to see more detail about each specific log record.

TBL

The name of the table associated with the transaction.

DBID

The database ID associated with the transaction.

ADD

The total number of add requests for a transaction.

DELETE

The total number of delete requests for a transaction.

UPDATE

The total number of update requests for a transaction.

Request Summary Report

The REQUEST SUMMARY REPORT in an RXX Analysis Report provides counts of adds, deletes, and updates to every table and base for the transaction, as shown in the following.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                             Multi-User Facility                              *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX
          REQUEST SUMMARY REPORT
     COMMAND  TABLE DBID AREA  OCCURRENCES PERCENT ❶

     CLOSE                            1      0.0
     COMIT                           48      0.0
     ROLBK                            6      0.0
     ADDIT    C01    997 A01       18,000     17.2
     DELET    C01    997 A01        8,000      7.6
     UPDAT    C01    997 A01        8,000     10.0
     ADDIT    C02    997 A01       11,328     10.8
     DELET    C02    997 A01       11,355     10.9
❷ MOVERU   C02    997 A01        1,056      1.0
     UPDAT    C02    997 A01       11,355     14.2

❸ ADDIT    TOTAL               38,034     36.4
     DELET    TOTAL               28,115     27.0
     UPDAT    TOTAL               28,056     35.1
```

❶

A PERCENT column was added in Version 12.0 for each detail line. The percentage is the percent of the occurrence records for the entire RXX input. Restrictive options like DBID= do not change the percent.

❷

The following pertains to the MOVER command:

MOVERU

MOVERU is specified for MOVER log records that were done for a record moved in support of the UPDAT command for compressed tables.

MOVERO

MOVERO is specified for MOVER log records that were done for a record moved in support of the DBUTLTY OLREORG function.

❸

In addition to the percentage, another Version 12.0 enhancement is another summary section consisting of one line for all ADDIT commands, one line for all UPDAT commands, and one line for all DELET commands. The information represents the occurrences of those commands for all selected records, and it is the percent of this summation.

Additional Summary Information

The next page in an RXX Analysis Report provides more summary information as follows.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                             Multi-User Facility                              *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

INPUT BLOCKS READ: --------269
TOTAL RECORDS SELECTED: ------1,020
DATE/TIME RANGE FROM mm/dd/ccyy hh.mm.ss TO mm/dd/ccyy hh.mm.ss
TSN RANGE FROM 00001A0E TO 000022E4
```

INPUT BLOCKS READ:

Includes the full input unless selection criteria allows processing to stop without passing all of the data.

TOTAL RECORDS SELECTED:

Counts the log records matching selection data (in this sample, all rows match and are selected).

DATE/TIME RANGE FROM

Provides the lowest date/time and the highest date/time of the input. It is not provided if the full input is not read because of selection options.

TSN RANGE FROM

Provides the lowest and highest transaction sequence number(s) found on the input. This information is not provided if the full input is not read.

Note: Some of the commands printed in the summary report are strictly internal to CA Datacom/DB and so their exact purpose is not described here. They are included for completeness and in case they are required for use by Technical Support.

Warning Messages

Warning messages are built into the RXX Analysis Report to alert you to unexpected data. The warnings do not necessarily represent a problem but could. The input is expected to be a series of log records in the order they were created. If this is found to not be true, a warning message is produced. For example:

```
WARNING:  TIMESTAMP STEP DOWN: FROM B229FA165F111C00 TO B229EC970BCCFB04 LXX BLOCK xxxxxxxx VS yyyyyyyy

WARNING:  DATE/TIME STEP DOWN: FROM mm/dd/ccyy hh.mm.ss TO mm/dd/ccyy hh.mm.ss LXX BLOCK xxxxxxxx VS yyyyyyyy

WARNING:  TSN STEP DOWN: FROM DBDB2706 TO DBDB2701 LXX BLOCK xxxxxxxx VS yyyyyyyy

WARNING:  LOG RECORD SEQUENCE NUMBER NOT +1: FROM 0000000F TO 0000005F LXX BLOCK xxxxxxxx VS yyyyyyyy
```

For the LXX BLOCK info in the previously shown example:

xxxxxxxx

Shows (for debugging purposes) the hex value for the log block prior to the current one.

yyyyyyyy

Shows (for debugging purposes) the hex value for the current log block.

Each of the four kinds of warning messages shown previously is independent of the others and occurs as often as its condition applies. For example, if the volumes of a multivolume file are provided in an order other than that in which they were built, one of each message would be expected.

Because an RXX Analysis Report can be lengthy and the warning message(s) can be embedded with the output (making it easy for them to be overlooked), at the bottom of the report (if any of the warning messages were generated) is a warning that alerts you to the embedded warning(s). The messages that warn you about the warnings are:

```
WARNING:  TIMESTAMP STEP DOWN(S) ENCOUNTERED.

WARNING:  DATE/TIME STEP DOWN(S) ENCOUNTERED.

WARNING:  TSN STEP DOWN(S) ENCOUNTERED.

WARNING:  LOG RECORD SEQ. NUMBER NOT +1(S) ENCOUNTERED.
```

Split File Analysis Report

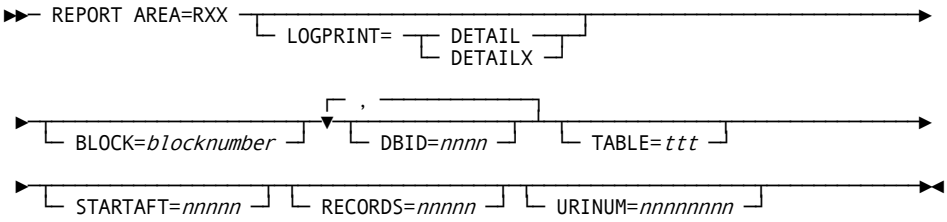
The Recovery File Analysis Report program can detect the type of input file. If the input is a CA Datacom Fast Restore file (split), the report control statements are as shown in the following syntax diagram.

The two kinds of example split file reports discussed in this section are:

- LOGPRINT=DETAIL Example Split File Report (see LOGPRINT=DETAIL Example Split File Report (see page 645))
- LOGPRINT=DETAILX Example Split File Report (see LOGPRINT=DETAILX Example Split File Report (see page 647))

How to Use

Following is the syntax for the RXX Split File Analysis Report.



Each of the following options can only be specified once, except for the DBID= option, that you can repeat.

Command

REPORT

Invokes the function to produce a report.

Required Keywords

AREA=RXX

Requests the Recovery File Analysis Report.

Optional Keywords

BLOCK=

Specify an eight-character hexadecimal number representing a data block within a data area. This option would normally be done as a request of Technical Support.

DBID=

Use to specify a database ID. You can repeat this option. Each occurrence may specify one or more valid database IDs or a range of DBIDs nnnn-mmmm, such as 100-250.

LOGPRINT=

Choose either DETAIL or DETAILX.

LOGPRINT=DETAIL is optional, that is, specifying DETAIL gives the same results as leaving a LOGPRINT= specification off and using only REPORT AREA=RXX (the basic syntax without options). For an example report generated by the basic syntax, see LOGPRINT=DETAIL Example Split File Report (see page 645).

LOGPRINT=DETAILX prints the entire record in hexadecimal. For an example report generated with LOGPRINT=DETAILX, see LOGPRINT=DETAILX Example Split File Report (see page 647).

RECORDS=

Provides a count of log records that are acceptable after processing the number of log records specified in the STARTAFT= option. If the STARTAFT= option is not provided, STARTAFT=0 is assumed. Specifying both RECORDS= and STARTAFT= gives you the ability to print/dump portions of a Recovery File (RXX).

Note: The input is not read beyond the number of records to be selected.

STARTAFT=

Specify a count of log records before which all are rejected.

TABLE=

Specify a table name.

URINUM=

Provides an eight-character URI number.

This utility always reads one input file of one or more volumes. In z/OS, this can be multiple concatenated files that appear to CA Datacom/DB as one file. If the input is not log data, unexpected results occur.

LOGPRINT=DETAIL Example Split File Report

Following is an example report generated by specifying LOGPRINT=DETAIL (same as the basic syntax without options, that is, REPORT AREA=RXX).

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
         ❶           REPORT AREA=RXX

                    FUNCTION=REPORT
         ❷               AREA=RXX


         ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************
                              ❶              ❷                    ❸
 SPLIT FILE       GROUP         GROUP START              GROUP END
                  GROUP9     1/05/00,10.40.44         1/05/00,10.41.45
          ❹      ❺      ❻          ❼         ❽
       DBID   TABLE    URI       BLOCK     ACTIVITY
        789    C01   00000001   00000002   A U D
        789    C01   00000002   00000002   A U D
        789    C01   00000003   00000002   A U D
        789    C01   00000004   00000002   A U D
        789    C01   00000005   00000002   A U D
        789    C01   00000006   00000002   A U D
        789    C01   00000007   00000002   A U D
        789    C01   00000008   00000002   A U D
        789    C01   00000009   00000002   A U D
                       .
                       .
                       .
```

❶

GROUP identifies the split file group.

❷

GROUP START gives the date/time the group started.

❸

GROUP END gives the date/time the group ended.

❹

DBID specifies a database ID. This is left blank for records without database IDs. It is printed as a decimal number.

❺

TABLE specifies a database 3-character table name.

❻

URI specifies a Unique Row Indentifier. This is left blank for records without a URI number. It is printed in hexadecimal format.

❼

BLOCK specifies a data area block number. This is left blank for records without a data block. It is printed in hexadecimal format.

❽

ACTIVITY lists the action taken on this record. These letters (action codes) can occur in any order. They are shown only when required/desired to support the CA Datacom/DB facilities (that is, they are provided for Technical Support and are not always shown, even though the description would indicate that it should be shown). The possible letters that can be shown on the report are:

A

Add

D

Delete

M

Moved

U

Update

R

Rollback

I

Incomplete task

The previous example defines the provided and optional reports. Each was described as though the entire input was considered selected data (the default), but be aware that numerous data selection options are available. Remember that a log record, to be selected, must pass all of the specified options (a single option can be specified or any combination, including all of them).

LOGPRINT=DETAILX Example Split File Report

Following is an example report generated by specifying LOGPRINT=DETAILX to give a hexadecimal dump of the entire record.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                 CA Datacom/DB                                  *
Time: hh.mm.ss      *                               General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                     *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
          ❶          REPORT AREA=RXX,LOGPRINT=DETAILX

                    FUNCTION=REPORT
          ❷             AREA=RXX
                        LOGPRINT=DETAILX


          ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************
                   ❶              ❷                    ❸
 SPLIT FILE      GROUP         GROUP START              GROUP END
                 GROUP9     1/05/00,10.40.44          1/05/00,10.41.45
  0000  00678001 C3C161C4 C1E3C1C3 D6D461C4   C261C6C1 E2E340D9 C5E2E3D6 D9C540C6     *    CA/DATACOM/DB/FAST RESTORE F*
  0020  C9D3C540 C7D9D6E4 D7F900B3 763E857B   4AE10000 01051040 44000105 10414500     *ILE GROUP9                         *
  0040  00000000 00000000 00000000 000000B3   66F16C90 344840B3 66F247CB 33518000     *                 1       2       *
  0060  00000000 000000                                                              *                                 *
        ❹    ❺     ❻          ❼        ❽
      DBID  TABLE    URI       BLOCK     ACTIVITY
       789   C01   00000001   00000002   A U D
  0000  00200700 0315C1F0 F1000000 02000000   0001C3F0 F1000000 02000000 00000000     *      A01         C01           *
       789   C01   00000002   00000002   A U D
  0000  00200700 0315C1F0 F1000000 02000000   0002C3F0 F1000000 02000000 00000000     *      A01         C01           *
       789   C01   00000003   00000002   A U D
  0000  00200700 0315C1F0 F1000000 02000000   0003C3F0 F1000000 02000000 00000000     *      A01         C01           *
                      .
                      .
                      .
```

❶

GROUP identifies the split file group.

❷

GROUP START gives the date/time the group started.

❸

GROUP END gives the date/time the group ended.

❹

DBID specifies a database ID. This is left blank for records without database IDs. It is printed as a decimal number.

❺

TABLE specifies a database three-character table name.

❻

URI specifies a Unique Row Indentifier. This is left blank for records without a URI number. It is printed in hexadecimal format.

❼

BLOCK specifies a data area block number. This is left blank for records without a data block. It is printed in hexadecimal format.

❽

ACTIVITY lists the action taken on this record. These letters (action codes) can occur in any order. They are shown only when required/desired to support the CA Datacom/DB facilities (that is, they are provided for Technical Support and are not always shown, even though the description would indicate that it should be shown). The possible letters that can be shown on the report are:

A

Add

D

Delete

M

Moved

U

Update

R

Rollback

I

Incomplete task

Data Area Space Estimation Report (Device)

The DEVICE= keyword of the REPORT function provides, for a given device type, the option of an estimate of the logical records per block and the total space required to store a particular number of records in a CA Datacom/DB data area.

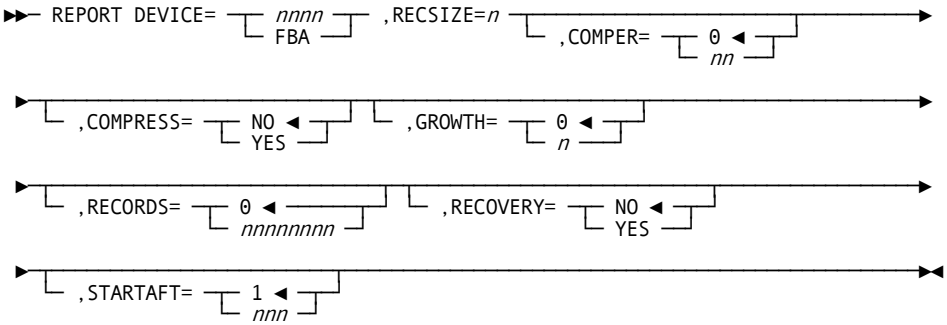
When to Use

Use the Data Area Space Estimation Report when planning the necessary allocation for a CA Datacom/DB data area.

You can request the REPORT DEVICE option at any time. See the *CA Datacom/DB Database and System Administration Guide* for the BLOCK-SIZE= area attribute requirements for restrictions regarding CA Datacom/DB.

How to Use

You can execute this command in either Single User or Multi-User mode. To obtain a Data Area Space Estimation Report, execute the REPORT function with the following command format.



Command

REPORT

Invokes the function to produce a report.

Required Keywords

DEVICE=

Specifies the device type for the report.

Valid Entries:

3375, 3380, 3390, 9345, or FBA

Default Value:

(No default)

,RECSIZE=

Specifies the CA Datacom/DB table record size.

Do not specify a value for RECSIZE= that is greater than the track size of the device selected. This value should not include the CA Datacom/DB Record Control Element (RCE).

Valid Entries:

1 to 32720

Default Value:

(No default)

Optional Keywords

,COMPER=

Specifies the estimated compression percentage. COMPRESS=YES must also be specified.

Valid Entries:

0 to 99

Default Value:

0

,COMPRESS=

Specifies whether compression is to be considered in estimating space.

Valid Entries:

NO or YES

Default Value:

NO

,GROWTH=

Specifies a growth percentage. The report shows space that should be allocated for growth, if this value is provided.

Valid Entries:

0 to 999

Default Value:

0

,RECORDS=

Specifies the estimated number of records to be stored in the table. If no value is supplied, the report only displays expected records per block and track.

Valid Entries:

0 to 99999999

Default Value:

0

,RECOVERY=

Specifies whether extra data needed for RECOVERY on each record is to be considered.

Valid Entries:

NO or YES

Default Value:

NO

,STARTAFT=

Specifies the first records-per-block value to be displayed on the space estimation chart.

Valid Entries:

1 to 999

Default Value:

1

Example JCL (3380)

The following shows the command to print the estimated space calculations for a 400-byte record stored on a 3380 device.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
         REPORT   DEVICE=3380,RECSIZE=400,COMPER=25,COMPRESS=YES,
                   GROWTH=25,RECORDS=250000,RECOVERY=YES
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT DEVICE (3380)

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶            REPORT  DEVICE=3380,RECSIZE=400,COMPER=25,COMPRESS=YES,
                                GROWTH=25,RECORDS=250000,RECOVERY=YES

                    FUNCTION=REPORT
       ❷               COMPER=25
                       COMPRESS=YES
                       DEVICE=3380
                       GROWTH=025
                       RECORDS=00250000
                       RECOVERY=YES
                       RECSIZE=00400

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing are presented in this area.

```
Date: mm/dd/ccyy    ********************************************************************************          Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

MATRIX FOR DEVICE TYPE....."3380".

        { ------------------------------------------------------------
        { #    47476  #    23476  #    15476  #    11476  #     9076  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     7476  #     6356  #     5492  #     4820  #     4276  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     3860  #     3476  #     3188  #     2932  #     2676  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     2484  #     2324  #     2164  #     2004  #     1876  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     1780  #     1684  #     1588  #     1492  #     1396  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     1332  #     1268  #     1204  #     1140  #     1076  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     1044  #      980  #      948  #      916  #      852  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      820  #      788  #      756  #      724  #      692  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      660  #      628  #      596  #      596  #      564  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      532  #      500  #      500  #      468  #      436  #
    ❹    { #-----------#-----------#-----------#-----------#-----------#
        { #      436  #      404  #      404  #      372  #      372  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      340  #      340  #      308  #      308  #      276  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      276  #      276  #      244  #      244  #      244  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      212  #      212  #      212  #      180  #      180  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      180  #      148  #      148  #      148  #      116  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #      116  #      116  #      116  #       84  #       84  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #       84  #       84  #       84  #       52  #       52  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #       52  #       52  #       52  #       20  #       20  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #       20  #       20  #       20  #       12  #       12  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #       12  #       12  #       12  #       12  #       12  #
        { #-----------#-----------#-----------#-----------#-----------#
```

❹

Matrix of block sizes for different blocks per track. The first row shows block sizes for 1 to 5 blocks per track. The second row shows 6 to 10 blocks per track, and so forth.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


-------------------------------------------------------------------------
# DEVICE # RECORD # RECOVERY #  COMPRESSION  # FIRST #  RECORD  #  GROWTH # }
#  TYPE  #  SIZE  #   Y/N    # Y/N # PERCENT # R P B # QUANTITY # PERCENT # }
#--------#--------#----------#-----#---------#-------#----------#---------# }  ❺
#  3380  #    400 #    Y     #  Y  #    25 % #    0  #   250000 #    25 % # }
-------------------------------------------------------------------------
-------------------------------------------------------------------------------------------
# RCD #  MINM # DEVICE #    LOGICAL    #           DATA SPACE REQUIRED               # }
# PER # BLOCK #  BLOCK #  RECORDS PER  # PRIMARY #  GROWTH #   TOTAL ##     TOTAL     # }
# BLK #  SIZE #   SIZE # TRACK # CYLIN #  TRACKS #  TRACKS #  TRACKS ##  CYLIN  / TRK # }
#-----#-------#--------#-------#-------#---------#---------#---------##---------/-----# }
#   1 #   314 #    340 #    57 #   855 #    4386 #    1097 #    5483 ##     365 /   8 # }
#   2 #   626 #    628 #    84 #  1260 #    2977 #     745 #    3722 ##     248 /   2 # }
#   3 #   938 #    948 #    99 #  1485 #    2526 #     632 #    3158 ##     210 /   8 # }  ❻
#   4 #  1250 #   1268 #   108 #  1620 #    2315 #     579 #    2894 ##     192 /  14 # }
#   5 #  1562 #   1588 #   115 #  1725 #    2174 #     544 #    2718 ##     181 /   3 # }
#   6 #  1874 #   1876 #   120 #  1800 #    2084 #     521 #    2605 ##     173 /  10 # }
#   7 #  2186 #   2324 #   119 #  1785 #    2101 #     526 #    2627 ##     175 /   2 # }
#   8 #  2498 #   2676 #   120 #  1800 #    2084 #     521 #    2605 ##     173 /  10 # }
    .      .        .       .       .          .        .         .          .      .   }
    .      .        .       .       .          .        .         .          .      .
    .      .        .       .       .          .        .         .          .      .
```

❺

Displays the parameters you coded with the REPORT DEVICE option. The report also displays the default values for parameters not coded.

❻

This table provides you with the information you need to determine optimum device block size. Use TOTAL TRACKS under the DATA SPACE REQUIRED column, locate the lowest value or the least amount of space to allocate. Read across the report to determine the DEVICE BLOCK SIZE. Other information is provided on the report which may be useful in making space allocation decisions or in determining whether to rerun the report with different parameter values.

Example JCL (FBA)

The following shows the command to print the estimated space calculations for a 2048 byte record stored on an FBA device.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
         REPORT   DEVICE=FBA,RECSIZE=2048,COMPER=25,COMPRESS=YES,
                   GROWTH=25,RECORDS=250000,RECOVERY=YES
 /*
 /&
 * $$ EOJ
```

Sample Report: REPORT DEVICE (FBA)

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶            REPORT  DEVICE=FBA,RECSIZE=2048,COMPER=25,COMPRESS=YES,
                              GROWTH=25,RECORDS=250000,RECOVERY=YES

                    FUNCTION=REPORT
       ❷               COMPER=25
                       COMPRESS=YES
                       DEVICE=FBA
                       GROWTH=025
                       RECORDS=00250000
                       RECOVERY=YES
                       RECSIZE=02048

       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing are presented in this area.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

MATRIX FOR DEVICE TYPE....."FBA ".
        { ------------------------------------------------------------
        { #    50176  #    49664  #    49152  #    48640  #    48128  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    47616  #    47104  #    46592  #    46080  #    45568  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    45056  #    44544  #    44032  #    43520  #    43008  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    42496  #    41984  #    41472  #    40960  #    40448  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    39936  #    39424  #    38912  #    38400  #    37888  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    37376  #    36864  #    36352  #    35840  #    35328  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    34816  #    34304  #    33792  #    33280  #    32768  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    32256  #    31744  #    31232  #    30720  #    30208  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    29696  #    29184  #    28672  #    28160  #    27648  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    27136  #    26624  #    26112  #    25600  #    25088  #
        { #-----------#-----------#-----------#-----------#-----------#
    ❹    { #    24576  #    24064  #    23552  #    23040  #    22528  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    22016  #    21504  #    20992  #    20480  #    19968  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    19456  #    18944  #    18432  #    17920  #    17408  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    16896  #    16384  #    15872  #    15360  #    14848  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    14336  #    13824  #    13312  #    12800  #    12288  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #    11776  #    11264  #    10752  #    10240  #     9728  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     9216  #     8704  #     8192  #     7680  #     7168  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     6656  #     6144  #     5632  #     5120  #     4608  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     4096  #     3584  #     3072  #     2560  #     2048  #
        { #-----------#-----------#-----------#-----------#-----------#
        { #     1536  #     1024  #      512  #        0  #        0  #
        { #-----------#-----------#-----------#-----------#-----------#
```

❹

Matrix of blocksizes for different blocks per track. The first row shows block sizes for one through five blocks per track. The second row shows six through ten blocks per track, and so forth.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


-------------------------------------------------------------------------
# DEVICE # RECORD # RECOVERY #  COMPRESSION  # FIRST #  RECORD  #  GROWTH # }
#  TYPE  #  SIZE  #   Y/N    # Y/N # PERCENT # R P B # QUANTITY # PERCENT # }
#--------#--------#----------#-----#---------#-------#----------#---------# }  ❺
#  FBA   #   2048 #    Y     #  Y  #    25 % #    0  #   250000 #    25 % # }
-------------------------------------------------------------------------
-------------------------------------------------------------------------
# RCD #  MINM # DEVICE #  FBA  BLOCKS  #     DATA SPACE REQUIRED     ## }
# PER # BLOCK #  BLOCK # REQUIRED  PER # PRIMARY #  GROWTH #   TOTAL ## }
# BLK #  SIZE #   SIZE # PHYSICAL  BLK #  BLOCK  #   BLOCK #   BLOCK ## }
#-----#-------#--------#--------------#---------#---------#---------## }
#   1 #  1559 #   2048 #         4     #  250000 #   62500 #  312500 ## }
#   2 #  3116 #   3584 #         7     #  125000 #   31250 #  156250 ## }
#   3 #  4673 #   5120 #        10     #   83334 #   20834 #  104168 ## }
#   4 #  6230 #   6656 #        13     #   62500 #   15625 #   78125 ## }
#   5 #  7787 #   8192 #        16     #   50000 #   12500 #   62500 ## }
#   6 #  9344 #   9728 #        19     #   41667 #   10417 #   52084 ## }  ❻
#   7 # 10901 #  11264 #        22     #   35715 #    8929 #   44644 ## }
#   8 # 12458 #  12800 #        25     #   31250 #    7813 #   39063 ## }
#   9 # 14015 #  14336 #        28     #   27778 #    6945 #   34723 ## }
#  10 # 15572 #  15872 #        31     #   25000 #    6250 #   31250 ## }
#  11 # 17129 #  17408 #        34     #   22728 #    5682 #   28410 ## }
#  12 # 18686 #  18944 #        37     #   20834 #    5209 #   26043 ## }
#  13 # 20243 #  20480 #        40     #   19231 #    4808 #   24039 ## }
    .     .        .             .           .          .        .      }
    .     .        .             .           .          .        .
    .     .        .             .           .          .        .
```

❺

Displays the parameters you coded with the REPORT DEVICE option. The report also displays the default values for parameters not coded.

❻

This table provides you with the information you need to determine optimum device block size. Use TOTAL BLOCK under the DATA SPACE REQUIRED column, locate the lowest value or the least amount of space to allocate. Read across the report to determine the DEVICE BLOCK SIZE. Other information is provided on the report which can be useful in making space allocation decisions or in determining whether to rerun the report with different parameter values.

Important! The MEMORY=CF report is provided for use by Technical Support and should *only* be executed if requested by them. They are able to interpret the results as necessary. No report is possible against the lock structure.

Type H - Utility Function History Report

The Utility Function History Report provides information about prior utility functions executed against the data sets in a database.

The INIT function on an Index Area or data area clears all information prior to the execution of the INIT function.

Note: Starting in Version 11.0, the entries for physical backups or EXTRACTs without UPDATE=YES have been removed from this report (running without UPDATE=YES does not have the database open and so may not issue a write to the data area control block to reflect that it occurred).

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

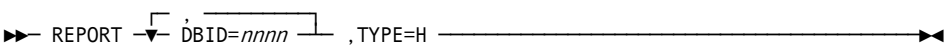
When to Use

Use the information provided by this report to determine when an area was backed up and with what choices. The report also lists the last dates when the RETIX or RESET CXX functions or the Restart process of the Multi-User Facility enable were executed.

This function executes through the Multi-User Facility and, therefore, the Multi-User Facility must be enabled. All tables must be loaded with a valid index because each area is opened with read status.

How to Use

Multi-User mode is required to execute this command. The following is the control statement to request this report:



Command

REPORT

Invokes the function to produce a report.

Required Keywords

DBID=

Identifies the database or databases to be reported. Each DBID= occurrence may specify one or more valid database IDs or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250).

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database not defined within a range is ignored.

Valid Entries:

One or more valid DATACOM-IDs of the database or a range of DBIDs as already described.

Default Value:

(No default)

,TYPE=H

Specifies the Utility Function History Report.

Example JCL

The following shows the command to generate a Utility Function History Report.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        REPORT   DBID=1,TYPE=H
 /*
 /&
 * $$ EOJ
```

Sample Report: Utility Function History

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             REPORT DBID=1,TYPE=H

                    FUNCTION=REPORT
       ❷               DBID=1
                       TYPE=H


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing would be displayed in this area.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                             General Utility                                  *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX

                    FUNCTION HISTORY REPORT

  ❶      ❷            ❸          ❹          ❺         ❻         ❼       ❽

AREA    EVENT         JOB NAME     DATE       TIME     CXXNAME    TYPE   OTHER

IXX     INIT          DBDVL     12/03/2002  09.36.41   PRODCXX
        EXTEND                                                          EXTENDS SINCE INIT -   0
DEM     INIT          DBDVL     12/03/2002  09.36.39   PRODCXX
        EXTEND                                                          EXTENDS SINCE INIT -   0
        RESTART
        LOAD          DBDVL     12/03/2002  09.36.41   PRODCXX    B
        LOAD (-1)     DBDVL     12/03/2002  09.36.39              B
        RETIX
        RETIX (-1)
        BACKUP
        BACKUP (-1)
        BACKUP (-2)
        BACKUP (-3)
        BACKUP (-4)
        BACKUP (-5)
        BACKUP (-6)
        BACKUP (-7)
PAY     INIT          DBDVL     11/07/2002  08.09.49   PRODCXX
        EXTEND                                                          EXTENDS SINCE INIT -   0
        RESTART
        LOAD          DBDVL     12/11/2002  09.00.00   PRODCXX
        LOAD (-1)     DBDVL     12/11/2002  08.00.00
        RETIX         DBDVL     11/14/2002  13.48.31
        RETIX (-1)
        BACKUP        DBDVL     12/11/2002  09.14.03              E
        BACKUP (-1)   DBDVL     12/11/2002  08.52.30              E
        BACKUP (-2)   DBDVL     11/07/2002  15.30.33              N
        BACKUP (-3)   DBDVL     11/07/2002  08.11.37              N
        BACKUP (-4)
        BACKUP (-5)
        BACKUP (-6)
        BACKUP (-7)
PMF     INIT          DBDVL     11/07/2002  08.09.50   PRODCXX
        EXTEND                                                          EXTENDS SINCE INIT -   0
        RESTART
        LOAD          DBDVL     12/03/2002  09.36.41   PRODCXX    B
        LOAD (-1)     DBDVL     12/03/2002  09.36.00              B
        RETIX
        RETIX (-1)
        BACKUP        DBDVL     11/07/2002  15.30.33              N
        BACKUP (-1)   DBDVL     11/07/2002  08.11.37              N
```

❶

AREA - The area with the data area name or IXX for the Index Area. If the database has Multi-Dataset Index areas, the index areas named I*nn* follow the IXX area and are in front of the data areas.

❷

EVENT - A list of the functions that have executed against the area.

❸

JOB NAME - The name of the job that executed the function.

❹

DATE -The date the function was executed.

❺

TIME - The time the function was executed.

❻

CXXNAME - The name of the Directory at the time the function executed. The CXXNAME is only available for the INIT and the last LOAD.

❼

TYPE - The choices made with the execution of the function (EVENT). The following shows the function that shows under EVENT, the value that shows under TYPE, and a description of the code.

BACKUP

Codes and descriptions

E

An EXTRACT was performed. Whether it was PHYSICAL or NATIVE is not provided.

N

SEQ=NATIVE

P

SEQ=PHYSICAL

R

SEQ=PHYSICAL, RECID=YES

EXTEND

Not applicable.

INIT

Not applicable.

LOAD

Codes and descriptions

0

FORMAT=NONE

E

FORMAT=EXTRACT

V

FORMAT=VAR

B

FORMAT=BACKUP (without control information)

N

FORMAT=BACKUP, SEQ=NATIVE

P

FORMAT=BACKUP, SEQ=PHYSICAL

R

FORMAT=BACKUP, SEQ=PHYSICAL, RECID=YES

RESTART

Not applicable.

RETIX

Not applicable.

❽

OTHER - Additional information, such as the number of extents since the last INIT function and a description of the origin of the data used in a LOAD function. OTHER can contain SORT=0 if the last LOAD or RETIX had SORT=0 set (and it was not a LOAD FORMAT=NONE). This is to show the databases that were not last subject to LOAD and RETIX with a SORT value greater than 0 (which we recommend).

Chapter 29: RESET (Reset Area)

RESET CXX (Release Directory and Database Control)

The RESET function releases the Directory (CXX) and database control or resets the Log Area (LXX) to empty. The two functions are described separately following.

This function clears the update-user control count, read-only-user control count, and Directory (CXX) exclusive control indicator. The function forces subsequent CA Datacom/DB processing to assume no other CA Datacom/DB processing is active.

Important! Use this function only when no other CA Datacom/DB processing is active. If this function is run and CA Datacom/DB jobs are active, the processing that occurs after this point produces unpredictable results.

If a DBUTLTY RESET AREA=CXX,DBID=nnn is performed after a MUF failure and prior to MUF RESTART processing, the RESTART phase of the next Multi-User Facility start up ignores any log records that pertain to that DBID. Should this occur, your data integrity on this DBID may be compromised.

When to Use

Use this function if an abend has occurred during a DBUTLTY execution that has been or is being corrected, but only when no other CA Datacom/DB processing is active.

RESET AREA=CXX Processing

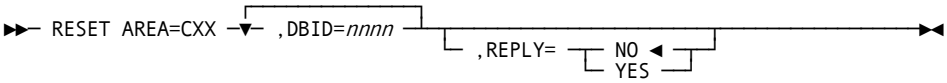
In processing a reset for the CXX, DBUTLTY attempts to connect to the Multi-User Facility. If successful, the Multi-User Facility is called to verify the base being reset is not open for update. If the base is open for update, the message DB13111E (see the *CA Datacom/DB Message Reference Guide*) is printed and the reset is rejected. If Multi-User is not enabled or cannot be checked for any reason, the reset continues.

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

How to Use

You can execute this command in either Single User or Multi-User mode. To perform a reset of database counters and the exclusive control indicator, execute the RESET function using the following command format:



Command

RESET

Invokes the function to release or reset an area.

Required Keywords

AREA=CXX

Identifies the area as the Directory.

,DBID=

Identifies the database to be released from control. DBID= may repeat, that is, each occurrence may specify one or more valid database IDs or a range of DBIDs *nnnn-mmmm* (a range, for example, such as 100-250). This function executes more efficiently when multiple entries are used.

Every database specified by a DBID keyword is processed. Each database that is not defined generates an error and fails the function after all the databases are processed, except that a database in a range not defined is ignored.

Valid Entries:

One or more valid DATACOM-IDs of the database or a range of DBIDs as already described.

Default Value:

(No default)

Optional Keywords

,REPLY=

Specifies the response required of the operator.

YES

Requires an operator response of RESET or BYPASS for each database entered (message DB13086A).

NO

Allows no operator action.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to reset the exclusive control on the Directory and all indicators for database 10.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
        RESET     AREA=CXX,DBID=10
 /*
 /&
 * $$ EOJ
```

Sample Report: RESET CXX

The RESET AREA=CXX report shows the OPEN status of databases as YES or NO.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                 Copyright © 2009 CA. All rights reserved.                    *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             RESET AREA=CXX,DBID=1

                    FUNCTION=RESET
       ❷               AREA=CXX
                       DBID=00001



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ******************************************************************************** Directory: PRODCXX

                    CONTROL FILE (CXX) OUTPUT -------------------------- EXCLUSIVE CONTROL NOT HELD
                                          ❹  STATUS  INDICATORS
                    BASE   OPEN UPDATE   OPEN READ        AREA   LOAD IN PROCESS   BACKUP IN PROCESS

                       1       YES          NO
                                                          DEM         NO                 NO
                                                          PAY         NO                 NO
                                                          PMF         NO                 NO
DB13087W - BASE 1 RESET, UPDATE USERS-YES, READ ONLY USERS-NO , OWNER-DBDVMUF  ❺
```

❹

Information provided by the STATUS INDICATORS.

❺

See message DB13087W in the *CA Datacom/DB Message Reference Guide* for more information.

RESET LXX (Reset Log Area to Empty)

This function of DBUTLTY updates the Log Area control record to indicate the Log Area is empty and ready for use.

When to Use

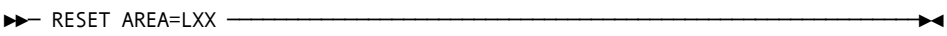
This function is not recommended for a production system. Use this function to enable a restart after the failure of a test system which can be readily rebuilt. On a production system, use the RESET LXX only in cases of dire emergency when the RESTART cannot be run.

Effect on Recovery

Using this function precludes using CA Datacom/DB recovery since the data that makes recovery possible is destroyed. For details on recovery, see RECOVERY (Rebuild a Database) (see page 443).

How to Use

You can execute this command in either Single User or Multi-User mode. To clear the Log Area, execute the RESET function using the following command format:



Command

RESET

Invokes the function to release or reset an area.

Required Keywords

AREA=LXX

Specifies the Log Area is to be reset.

Example JCL

The following shows the command to reset the Log Area to an empty condition.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL LXX,'lxx',,DA                         Log Data File
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
         RESET     AREA=LXX
 /*
 /&
 * $$ EOJ
```

Sample Report: RESET LXX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             RESET AREA=LXX

                    FUNCTION=RESET
       ❷                AREA=LXX



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *              SPnn
                    ******************************************************************************** Directory: PRODCXX


                                              LOG FILE (LXX) OUTPUT


DB13084I - LOG AREA RESET IS COMPLETE   ❹
```

❹

Message indicating the Log Area was successfully reset.

Chapter 30: RETIX (Reconstruct Index)

Use the RETIX function to rebuild:

- The data space index for one area of a database. (See below.)
- The Index Area (IXX) for one or all data areas of a database. (See Rebuilding an Index Area (see page 677).)

The RETIX function of DBUTLTY reads the data area in a physical sequence to rebuild an Index Area.

You can also use the RETIX function to reset the Transaction Sequence Number (TSN). (See Resetting the TSN (see page 688).)

See COMM STATUS (Print Status of Multi-User Jobs) (see page 180) for information about DBUTLTY STATUS console command support for this function.

For information about dynamic extend considerations with regard to this function, see DBUTLTY Dynamic Extend Considerations, and for information about the DYNAMIC_EXTEND console-like command, see the *CA Datacom/DB Database and System Administration Guide*.

Rebuilding the Data Space Index or Recounting Records

CA Datacom/DB uses the data space index to track the space available in a data area which is defined with space reclamation (data area space management options 1, 2, or 3). See the *CA Datacom/DB Database and System Administration Guide* for an explanation of the CA Datacom/DB data area space management options.

You can also use KEYNAME=*SETR with the RETIX function to recount records (for more information on *SETR, see Using KEYNAME=*SETR to Recount Records and Update CXX (see page 675)).

When to Use

Use this form of the RETIX function when you want to rebuild the data space index, freeing any held space in the data area without performing a full RETIX on the area.

Also use this function when you want to use KEYNAME=*SETR to recount records as described on page Using KEYNAME=*SETR to Recount Records and Update CXX (see page 675).

Any Multi-User Facility abend can result in lost data space information.

Transaction backout requires deleted records be saved, making that space unavailable for reuse until task commit. CA Datacom/DB removes this space from the supply of available space until it is freed by a later commit, DBUTLTY LOAD function, or DBUTLTY RETIX function. Therefore, you can use this form of the RETIX function to rebuild the data space index, freeing any held space in the data area without performing a full RETIX on the area.

Note: This function does not free space in the Index Area and does not affect the normal key usage Index Area.

Multi-User Facility Connection when KEYNAME=*DATA or KEYNAME=*SETR

This function communicates with one Multi-User Facility. You can use it with more than one Multi-User Facility when you specify KEYNAME=*DATA. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

Using KEYNAME=*SETR to Recount Records and Update CXX

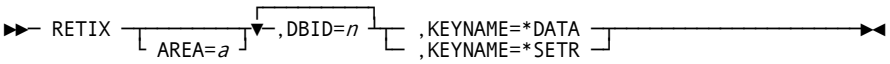
The count of records in a table, as known to the CXX, can become wrong if the MUF is restarted after a MUF outage occurs. The DBUTLTY RETIX function, with the KEYNAME=*SETR option, lets you recount the records and set the new value in MUF. Using the keywords DBID=, AREA=, and KEYNAME=*SETR with RETIX opens the database and reads through the area, using GSETP/GETPS commands to find all records. After processing all blocks, the count sets in the CXX definition. If no other task is concurrently running to do adds or deletes, at the completion of the RETIX function the count reflects the number of records. If another task is concurrently running, however, the new counts could be high, low, or equal, depending upon concurrency, but they will still almost certainly be more accurate than before the RETIX function with the KEYNAME=*SETR option was executed.

If the counts need to be updated on DASD, follow the RETIX function with a COMM STATS function.

Note: The RETIX function executes only for a URI area.

How to Use

When using the RETIX function to rebuild the data space index, the Multi-User Facility must be active. This utility function executes only in the Multi-User Facility. The database need not be disabled nor the User Requirements Tables closed. Use the following format for this function:



Command

RETIX

Invokes the function to rebuild a data space index or Index Area.

Optional Keyword

AREA=

Identifies the data area for which the data space index is to be rebuilt.

Valid Entries:

DATACOM-NAME of the area

Default Value:

(No default)

Required Keywords

,DBID=

Specifies the database containing the area for which the data space index is to be rebuilt. Each occurrence can specify one or more valid database IDs or a range of DBIDs *nnnn-mmmm.*

Valid Entries:

One or more valid database IDs or a range of DBIDs as already described

Default Value:

(No default)

,KEYNAME=

Use KEYNAME=*DATA to specify that the data space index be rebuilt. (for more information on *DATA, see Multi-User Facility Connection when KEYNAME=*DATA or KEYNAME=*SETR (see page 674)).

Use KEYNAME=*SETR to specify that records by recounted (for more information on *SETR, see Using KEYNAME=*SETR to Recount Records and Update CXX (see page 675)).

Valid Entries:

*DATA, *SETR

Default Value:

(No default)

Rebuilding an Index Area

You can also use the RETIX function to rebuild an Index Area which has been destroyed or damaged in some way. When using the RETIX function to rebuild the Index Area for all areas in a database at once, you must initialize the Index Area immediately before using the RETIX function.

When to Use

Use the RETIX function to rebuild an Index Area when you have added a new key definition to a table. If you redefine an existing key, making it longer or shorter, you must first remove it from the Index Area (IXX). See REMOVE (Remove Key ID): (see page 471) for details. Then, rebuild the Index Area using the RETIX function.

Note: When CA Datacom/DB encounters a duplicate value for a Unique Key or a Master Key that has been defined as allowing no duplicates (this can happen if the table is loaded with LOADDUPS=YES specified), CA Datacom/DB produces a snap dump of the duplicate. See the section on using dumps in the *CA Datacom/DB Database and System Administration Guide* for a sample.

After key definition changes, users normally execute the DBUTLTY RETIX function to rebuild the index for affected areas or the entire database. In previous releases, RETIX read the data area in physical sequence, extracted current key definitions, and merged them into the index in a process that also deleted any unnecessary index entries. By using the new RETIX keyword, MINIMAL=, you can restrict RETIX to only performing the minimal actions necessary.

The use of MINIMAL=YES is allowed at the area level only, not the database level. The AREA= keyword is therefore also required. The use of MINIMAL= usually reduces the number of I/O events, reduces the amount of CPU used, and reduces the elapsed time of the RETIX, compared to not using MINIMAL=. The savings are larger for areas with more records, compared to areas with fewer records. Areas with no records or few records might see a small saving or a small increase. A RETIX cannot run after the INIT of the IXX. It can run after the INIT of a Multi-Dataset Index other than the IXX, such as I02. Using MINIMAL= requires that the tables in the area all have their URI index set to loaded. The CXX REPORT must therefore not have an INDEX LOADED status of NO.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any Multi-User Facility for read.
- The database may *not* have any table in *unloading* status.

Environmental **Controls** *when MULTUSE=NO*

- The database will be opened for update.
- The utility has no knowledge about current ACCESS database or area status.
- The utility sets no ACCESS database or area status.

Environmental **Requirements** *when MULTUSE=YES*

- The database may or may not be open for update in the connected Multi-User Facility.
- The database may *not* be open for update in another Multi-User Facility or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The area may have no table in an open User Requirements Table.
- The area may *not* have any other MULTUSE=YES utility executing.
- The area must be closed.

Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by Multi-User Facility for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets RETIX executing this Multi-User Facility this database or area.

Special Rules with Database having a Multi-Dataset Index

A database with Multi-Dataset Index areas has special requirements that include the following:

- SORT must be specified as greater than 0 (zero).
- The data areas must be in the URI format.

Dynamic Extend during RETIX

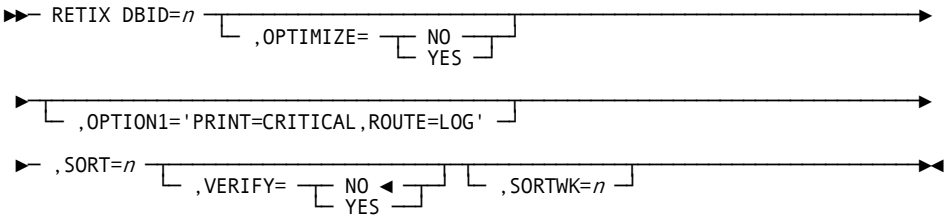
Dynamic extend of an index area is attempted if requirements are met. If requirements are not met, dynamic extend is not attempted. The requirements are as follows:

- SORT must be specified as greater than 0 (zero).
- The data areas must be using the URI option.

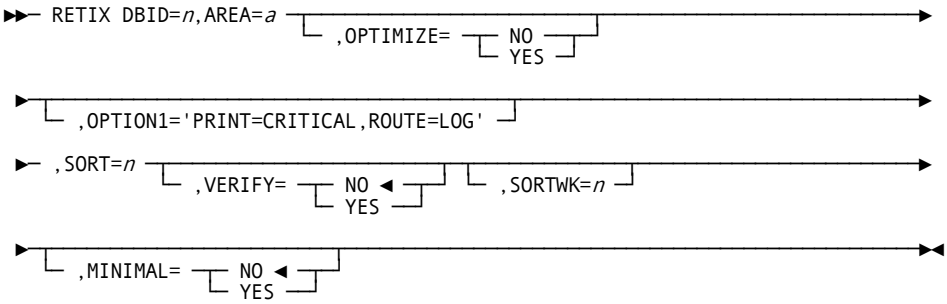
How to Use

You can execute this command in either Single User or Multi-User mode. Execute the RETIX function using the following command format.

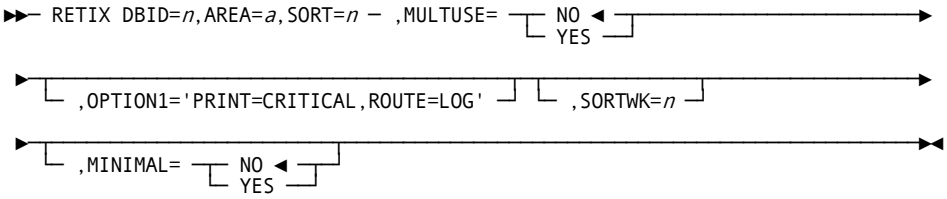
Rebuilding a Full Index Area



Rebuilding an Area's Index in Single User



Rebuild an Area's Index Area in Multi-User



Command

RETIX

Invokes the function to rebuild a data space index or Index Area.

Required Keywords

DBID=

Specifies the database for which the Index Area is to be rebuilt.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Optional Keywords

,AREA=

Identifies the data area or all areas in the database for which the Index Area is to be rebuilt. If this parameter is omitted, all areas are processed.

Valid Entries:

DATACOM-NAME of the area in the database

Default Value:

All areas are processed

MINIMAL=

Specifying MINIMAL=NO (the default) requires the RETIX function to perform everything it did in releases prior to Version 12.0. If MINIMAL=YES is specified, however, RETIX does only the minimal actions necessary to correct the key definition index for tables in the area.

Note: MINIMAL=YES requires SORT= to be specified with a value greater than 0 (zero).

For key definitions marked not-loaded, only those keys participate in the normal RETIX process.

For key entries in the index that have no current key definition, the low-level (DXX) blocks of the index are sequentially searched for pointers that need to be deleted.

MINIMAL=YES is only allowed at the area level (full database processing is invalid), and is only available for areas that are loaded in the URI format. When you specify MINIMAL=YES, make certain that every table in the area is loaded and has its URI index loaded.

Messages DB13283E is related to the use of the MINIMAL= keyword. For details about this message, see the *CA Datacom/DB Message Reference Guide*.

For information about changes in the RETIX report when MINIMAL= is used, see RETIX Report Changes Related to MINIMAL=.

Valid Entries:

YES or NO

Default Value:

NO

,MULTUSE=

Specifying MULTUSE=YES allows a single area in a database to have its index rebuilt, with no loss of read and update access to other areas in the database. When MULTUSE=YES, the function executes in a new format that is a combination of Multi-User Facility and Single User modes. The database involved must be able to be opened for update in the Multi-User Facility address space. The area, however, must have no tables open. If the database is currently open, CA Datacom/DB closes this area with the area close (you would need to close the area yourself *only if* the database is open and you need the Multi-User Facility to close it, so that any type of open data set lock can be released, which could be necessary in order to scratch and reallocate the data set). Once started, the function does not allow other tasks to open any tables in the area being processed. Other tables in the involved database, but not in the area, may be open for read or update in the same Multi-User Facility. With MULTUSE=YES:

- The area is first opened in Multi-User Facility for the RETIX.
- The area is also opened in Single User mode.
- The data area processing occurs in the utility address space.
- The key build/sort occurs in the utility address space.
- The index load/merge occurs in the Multi-User Facility address space. Packages of index entries (about 32k bytes) are sent from the utility to the Multi-User Facility.

When MULTUSE=YES, the SORT= option must be specified but not as zero (0). When MULTUSE=YES, the following options are not allowed to be specified: OPTIMIZE= and VERIFY=.

When MULTUSE=NO is specified or is allowed to default in the LOAD and RETIX functions, the functions execute as in versions prior to Version 11.0, that is, the database involved may not be open for update, and all processing executes in the Single User mode.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control (see page 43).

Valid Entries:

NO or YES

Default Value:

NO

,OPTIMIZE=

Specifies whether the Index entry creation optimization option is to be used. When YES is selected, if this Index Retrieve fails for any reason, reinitialize the Index Area. If any other data areas are already loaded, re-create the Index entries using this function of DBUTLTY, or reload all areas. If you select YES, you must also specify a value in the SORT= parameter.

NO

All Index entries are to be validated as they are created.

YES

All Index entries are not to be completed until the Index retrieve terminates. Selection of this option enhances performance by deferring Index pointer validation and complete Index structure update.

Note: Do not specify this option if MULTUSE=YES is specified.

Valid Entries:

NO or YES

Default Value:

NO (if one area is specified)
YES (if the full database is processed)

,OPTION1=

(z/VSE only) Specifies the options for the associated sort/merge program application. The values for this keyword must be 44 characters or less, and they must be enclosed in quotes. Depending on the features of your installations sort, the values that apply to this keyword can vary. CA Datacom/DB does not edit anything that you place inside of the quotes.

Valid Entries:

Any values that your installation's sort allows. These values must be less than 44 characters and be inside quotes.

Default Value:

CRITICAL for PRINT

LOG for ROUTE

,SORT=

Activates the Index Area entry sort option. When you specify a value, no Index Area entries are created as the data is retrieved. The required Index Area entries are sorted at the end of the data retrieval and merged into the Index Area in ascending sequence. Selecting this option enhances the performance of the Index Retrieve process for large databases. It is highly recommended when the number of records exceeds several hundred.

Note: The SORT option is required when specifying the RETIX function for the full database, or if you specify VERIFY=YES, OPTIMIZE=YES, or MULTUSE=YES.

Specify that the sort is to be used by entering a non-zero number. Any number from 1 to 999999999 is valid.

If sort is selected by entering a non-zero number, the Index entries in the Index Area (IXX) are not deleted by a separate Index pass. The Index Area is updated and only the Index blocks that reflect changes are written. If the SORT option is used, the partition size should be at least one megabyte.

Note: Do *not* specify over 99 million unless you verify that the sort package you are using allows a larger number.

Specifying SORT=0 (or any number of zeros) is the same as not specifying the keyword.

Many features introduced in previous releases require a non-zero value for SORT=, and the following two features, new for Version 12.0) that are also blocked during LOAD or RETIX executions when the sort option is not being used are as follows:

- on-demand dynamic extend of any data area or Index Area
- Execution against a database with Multi-Dataset Indexes

Most user-JCL is expected to already have a non-zero value for SORT= specified. If, however, a user has a LOAD or RETIX function coded in their JCL with SORT=0 specified or defaulted (SORT= not specified at all), beginning with Version 12.0 this information is added to the history report, and the first time in each MUF execution that an area is opened in a database that has a Version 12.0 LOAD or RETIX coded without the sort option, an informational message is generated. The message alerts you to the condition and can therefore help you know to first add the sort option, then see the results.

Two messages related to using LOAD and RETIX without the sort option are messages DB02809I and DB13128E. For details about these messages, see the *CA Datacom/DB Message Reference Guide*.

Valid Entries:

0 to 99999999999

Default Value:

(No default)

,SORTWK=

Specifies the number of sort work areas. The parameter is always edited, but is only used when the SORT= parameter is specified as not zero.

Note: Do not specify a 2-digit number unless the sort package you are using allows a 2-digit number. Sort packages that do not support a 2-digit number can fail in an unpredictable manner, if you specify a 2-digit number.

Valid Entries:

0 to 99

Note: Do not use two-digit numbers unless the SORT package to be used supports the larger number.

Default Value:

3

,VERIFY=

Verifies if the Index Area and data area are in exact agreement. It is only used when the SORT= parameter is specified as not zero.

Specify VERIFY=YES to activate the search for differences in the current Index and the Index being built. If a difference is detected, a Master List dump is produced. Only the first five differences cause a dump. Any additional differences are ignored.

The dumps have no effect on the execution of the RETIX function which continues to successful completion regardless of the value specified for the VERIFY option.

If you specify VERIFY=YES, you must specify the SORT option. If you do not specify SORT= when using VERIFY=YES, the first five records found cause a Master List dump whether or not a difference is found between the current Index and the Index being built.

Do not specify VERIFY=YES if you are using the RETIX function for the full database. Using the VERIFY is also *not* recommended for non-URI areas because records that have been moved are dumped, even though this condition is not an error.

Note: Do not specify this option if MULTUSE=YES is specified.

Valid Entries:

NO or YES

Default Value:

NO

Example JCL

The following shows the command to rebuild the Index for the DEM area on database 1.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL IXX001,'db001.Index',9999,DA               Index Area
 // EXTENT SYSnnn,volser,1,0,rel trk/blk,number trks/blks
 // DLBL DEM001,'db001.dem',,DA                     Data area
 // EXTENT SYSnnn,volser
 // ASSGN SYS001,device-type,VOL=volser,SHR
 // ASSGN SYS002,device-type,VOL=volser,SHR
 // ASSGN SYS003,device-type,VOL=volser,SHR
 // DLBL SORTWK1
 // EXTENT SYS001,volser,,,rel-trk/blk,number trks/blks
 // DLBL SORTWK2
 // EXTENT SYS002,volser,,,rel-trk/blk,number trks/blks
 // DLBL SORTWK3
 // EXTENT SYS003,volser,,,rel-trk/blk,number trks/blks
 // EXEC DBUTLTY,SIZE=750K
         RETIX     DBID=1,AREA=DEM,OPTIMIZE=YES,SORT=1000
 /*
 /&
 * $$ EOJ
```

Sample Report: RETIX

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             RETIX  AREA=DEM,DBID=1,OPTIMIZE=YES,SORT=1000

                    FUNCTION=RETIX
       ❷               AREA=DEM
                       DBID=1
                       OPTIMIZE=YES
                       SORT=00001000


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX

              TABLE          RECORDS
            { PNC                 0
            { PNM                 0
       ❹    { POH                 0
            { POL                 4
            {
            { TOTAL               4
```

❹

Confirmation that four records were indexed from the requested table.

RETIX Report Changes Related to MINIMAL=

The following RETIX report changes, related to the use of the MINIMAL= keyword, were made in Version 12.0.

If MINIMAL=YES is specified, the RETIX report starts with a standard RETIX report with table names and record counts, followed by a section showing tables and key names that have been built. These are the key definitions that were marked "not loaded or NL." If no table was found with a key of "not loaded," the word NONE appears to indicate that fact, as shown in the following sample report:

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     3
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                   Copyright © 2009 CA. All rights reserved.                  *               SPnn
Base:     1         ********************************************************************************* Directory: PRODCXX

              TABLE          RECORDS

              C01              6,000
              TOTAL            6,000

              TABLE       KEY NAME LOADED ❺
              C01              ANEWK
              c01              ZNEWK

              KEY ID DELETED PROCESSED ❻
              NONE
```

❺

KEY NAME LOADED is the key name, not in a loaded status, that was processed by this RETIX MINIMAL= option and is now loaded. All Keys that are not loaded participate in the RETIX MINIMAL=YES function.

In the sample report just shown, TABLE is the CA Datacom table that the key is in.

❻

KEY ID DELETED PROCESSED indicates that if there were keys in the index that were no longer being used, the RETIX MINIMAL= option deletes them and lists their KEYIDs here. The sample just shown lists NONE for this category, but see the following for a different case.

The following sample report shows key ID values that have been in the index (since the last INIT) and currently have no CXX definition (indicating they have been deleted). The key ID is the only value known for a former key. If no key IDs exist, the word NONE is used to indicate that fact, as shown in the previously given sample report.

In Single User (the default), all deleted keys are printed. In Multi-User, the first seven deleted keys are printed, then if more than seven exist, a note of the fact that there are more than seven is printed.

```
Date: mm/dd/ccyy    ********************************************************************************         Page:     4
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
Base:     1         ******************************************************************************** Directory: PRODCXX

               TABLE          RECORDS

               C01                 9
               TOTAL               9

               TABLE       KEY NAME LOADED
               C01              BNEWK
               c01              CNEWK

               KEY ID DELETED PROCESSED
                    8
                   10
```

Resetting the TSN

The Transaction Sequence Number (TSN), of the transaction that added or last updated each row, is stored in the Record Control Element (RCE) with each row. The TSN is used to ensure that the transaction has ended before granting a row lock to another transaction. A reset sets the TSN in the row to zero (that is never used as a TSN). The TSN is not normally reset because it does not normally need to be reset.

When to Use

The master TSN is stored in the Log Area (LXX). When the Log Area is initialized, the TSN number is set either to zero or to a value specified by the user. An error condition occurs when the Log Area is initialized with a lower TSN number than currently exists in the existing rows. This can be prevented by assigning a larger TSN number when initializing a Log Area. The error condition occurs because, if TSN numbers are reused, a new transaction could possibly wait on another executing transaction that has the same number as an already committed transaction. This would involve a short wait on the *wrong* transaction and is only a minor concern.

Multi-User Facility Connection when KEYNAME=*RTSN

This function communicates with one Multi-User Facility. You can use it with more than one Multi-User Facility when you specify KEYNAME=*RTSN. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

How to Use

You can execute this command in either Single User or Multi-User mode. When using the RETIX function to reset the TSNs, the Multi-User Facility must be active. This utility function executes only in the Multi-User Facility. The database need not be disabled nor the User Requirements Tables closed. Use the following format for this function:

Command

RETIX

Invokes the function to reset the TSN.

Required Keywords

AREA=

Identifies the data area for which the data space index is to be rebuilt.

Valid Entries:

DATACOM-NAME of the area

Default Value:

(No default)

,DBID=

Specifies the database containing the area for which the data space index is to be rebuilt.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

,KEYNAME=

Specifies that the TSN is to be reset to zero.

Valid Entries:

*RTSN

Default Value:

(No default)

Chapter 31: RXXFIX (Recover End-Of-File)

RXXFIX

The RXXFIX function reads a Recovery File and copies blocks to a new Recovery File up to a known stable point in time.

When to Use

Use the function to recover if the operating system or the Multi-User Facility terminates abnormally while spilling to an active Recovery File, leaving the tape with no End-of-File or trailer records.

RESTART Processing of a Tape Without End-of-File

If the tape has no End-of-File or trailer records, the RESTART process detects the condition and reports the incomplete cycle number. The RESTART process then changes the cycle and the log blocks being spilled from the status of "being spilled" to a status of "needing to be spilled." A later spill can then copy these blocks in their entirety.

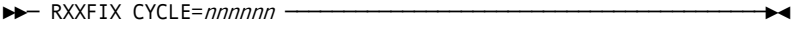
Two problem situations exist at this point:

- Because no trailer records exist on the active recovery tape, recovery processing with this tape is a problem. Backward recovery cannot function at all because the trailer records cannot be found. Forward recovery can function, but the recovery receives an I/O error.
- The second problem is that when executing forward recovery, the next Recovery File after the one with no End-of-File, detects errors because of the duplicated log records which require review.

The RXXFIX function eliminates these problems. By specifying a cycle number from the RESTART process, the utility copies all the good data to a new tape before any possible I/O error, stops reading the suspect tape, and closes the new complete, non-redundant tape, if a spill was ongoing at the time of the failure.

How to Use

Use the following command format to execute the RXXFIX function:



Command

RXXFIX

Invokes the function to read a Recovery File and copy blocks to a new Recovery File up to a known stable point in time.

Required Keyword

CYCLE=

Specifies the cycle number to be copied to. Provide the number supplied by the RESTART utility minus 1.

Valid Entries:

1 to 999999

Default Value:

(No default)

Example JCL

The following shows the command to ensure that you have a correct and usable Recovery File.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).


 // TLBL RXXIN,'rxxin.file'                      Recovery input file
 // ASSGN SYS005,TAPE
 // TLBL RXXOUT,'rxxout'                         Recovery output file
 // ASSGN SYS010,TAPE
 // EXEC DBUTLTY,SIZE=750K
         RXXFIX    CYCLE=5
 /*
 /&
 * $$ EOJ
```

Sample Report: RXXFIX

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             RXXFIX CYCLE=5

                    FUNCTION=RXXFIX
       ❷                CYCLE=000005



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                   { 0000001   CYCLE COPIED
                   { 0000002   CYCLE COPIED
               ❹   { 0000003   CYCLE COPIED
                   { 0000004   CYCLE COPIED
                   { 0000005   CYCLE COPIED
*** EOJ ***
```

❹

Messages indicating the status of the Recovery File.

Chapter 32: SECURITY RESET (Clearing Security Validation Buffer)

SECURITY RESET

The SECURITY RESET function is used to clear the security validation buffer. The SECURITY function requires that the Multi-User Facility be active. See the *CA Datacom Security Reference Guide* for information about security for CA Datacom/DB.

When to Use

Use the SECURITY RESET function to clear the security validation buffer. Security maintains a working set of validated profiles. This set is dynamically changed and is periodically cleared. If an access is changed, this reset must be done to ensure instant revalidation. This work set should be done only when changes have been made in your external security specifications that might be in the working set and, therefore, it is critical for the set to be refreshed.

This function communicates with one Multi-User Facility. Be aware of the following:

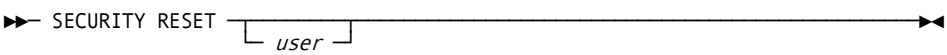
- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When using a console command, these steps are unnecessary.

How to Use the Console Command

Multi-User mode is required to execute this command.

Use the following in your command to initiate this function from the system console. Consult your site operating system documentation for additional syntax information.



Command

SECURITY

Invokes the function.

Required Keyword

RESET

Signs off all users and clears the security validation buffer.

Optional Keyword

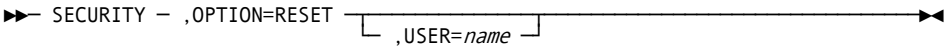
user

If the *user* parameter is not coded, SECURITY RESET signs off all users and the security validation buffer is cleared.

If the *user* parameter is coded, the individual user specified with the parameter is signed off and that individual's CICS, Server, and Gateway cached security validations are cleared. For active batch jobs, the individual user's batch cached security validations remain in effect for the life of the open URT for that batch job.

How to Use the DBUTLTY Command

Multi-User mode is required to execute this command. Execute the SECURITY RESET function with the following command format:



Command

SECURITY

Invokes the function.

Required Keywords

,OPTION=RESET

Signs off all users and clears the security validation buffer.

Default Value:

(No default)

Optional Keyword

,USER=*name*

If the USER= parameter is not coded, SECURITY RESET signs off all users and the security validation buffer is cleared.

If the USER= parameter is coded, the individual user specified is signed off and that individual's CICS, Server and Gateway cached security validations are cleared. For active batch jobs, the individual user's batch cached security validations remain in effect for the life of the open URT for that batch job.

Default Value:

(No default)

Example JCL

The following shows the command to execute the RESET option.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // EXEC DBUTLTY,SIZE=750K
          SECURITY  OPTION=RESET
 /*
 /&
 * $$ EOJ
```

Sample Report: SECURITY

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             SECURITY OPTION=RESET

                    FUNCTION=SECURITY
       ❷               OPTION=RESET



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

DB13105I - SECURITY RESET    REQUEST ACCEPTED    ❹
```

❹

Message indicating the success of the function execution.

Chapter 33: SET (Changing Error Processing)

SET

The SET function allows you to establish how you want DBUTLTY to support error handling of functions within a specific execution and for other special needs.

Two error handling issues of concern exist and each are enhanced with this function. The first is that if a function fails for any reason, each remaining function in the step is syntax edited and ignored. The second is that at the end of the last function, DBUTLTY ends with an abend 4 in z/OS or CANCEL in z/VSE and never a return code other than zero. These actions remain the default and cannot be globally changed.

The STOP and CONTINUE options indicate the execution option for future functions in the same utility step. The ABEND and RC*n* options indicate the actions taken to terminate the utility step back to the operating system. The STOP/CONTINUE options are completely independent from the ABEND/RC*n* options.

The SET function does not open any CA Datacom/DB tables.

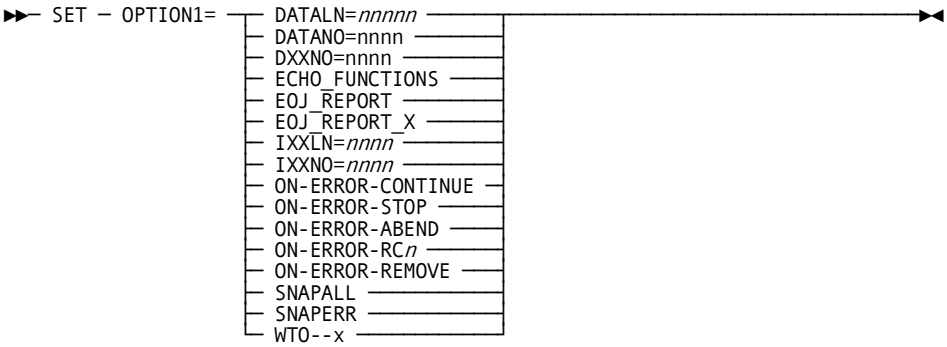
A SET function can generate an error condition only for a syntax error. If syntax is correct, it always executes except if the EDIT function was previously specified.

When to Use

Use this function when setting up JCL for a DBUTLTY function for the first time or when making changes to JCL.

How to Use

Multi-User mode is required to execute this command. To edit a stream of DBUTLTY functions, execute DBUTLTY using the following command format preceding the input stream to be edited:



Command

SET

Invokes the SET function for the input stream following this command.

Required Keywords

OPTION1=

Specifies the options for the SET function, as follows:

DATALN=*nnnnn*

The length of a data buffer as set in the DBMSTLST Master List assembly can be overridden, for any reason, by providing this option. The *nnnnn* must be replaced by a number in the range 4096-32767. This option is always accepted but only used if it occurs before the first function that opens the CXX.

DATANO=*nnnn*

The number of data buffers as set in the DBMSTLST Master List assembly can be overridden, for any reason, by providing this option. The *nnnn* must be replaced by a number in the range 3-9999. Some DBUTLTY functions need minimal data buffers and some would benefit with a large number. This option is always accepted but only used if it occurs before the first function that opens the CXX.

DXXNO=nnnn

The number of DXX buffers as set in the DBMSTLST Master List assembly can be overridden, for any reason, by providing this option. The *nnnn* must be replaced by a number in the range 3-9999. Some DBUTLTY functions need minimal data buffers and some would benefit with a large number. This option is always accepted but only used if it occurs before the first function that opens the CXX.

ECHO_FUNCTIONS

This option causes the first statement of each function to be sent to the console to aid in debugging or tracking the function being executed. This option can be helpful in seeing which other console messages occurred by which functions. Take note that that there is no equal sign and no additional values after the function name. It can be easier to see the function being executed in the log instead of going to the bottom of the print. The action once requested cannot be turned off. Following is an example:

```
DB10098I - STARTING FUNCTION - REPORT AREA=CXX,DBID=0997,TYPE=B
```

EOJ_REPORT

This option is intended to only be used in consultation with Technical Support. EOJ_REPORT requests that reports be generated at end of the current DBUTLTY execution, similar to the way in which MUF EOJ reports are generated. Information that is specific to the MUF is excluded from the report requested by EOJ_REPORT, for example Memory Resident Data Facility (MRDF) Summary Information reports and Task Control Block (TCB) Use Summary Information reports are excluded.

The EOJ_REPORT option includes a PXX-like summary report.

DBUTLTY functions use fast path processing and all statistical information normally present in the MUF may seem inconsistent. For example, a BACKUP shows I/O to the index and data areas but no requests to the tables. This result is not an error in DBUTLTY.

EOJ_REPORT_X

This option is intended to only be used in consultation with Technical Support. While the EOJ_REPORT option requests that reports be generated at end of the current DBUTLTY execution, the EOJ_REPORT_X option requests a report be generated at the end of the execution of most DBUTLTY functions. For information about what is produced, see the previously provided description of EOJ_REPORT.

IXXLN=nnnn

The length of an index buffer as set in the DBMSTLST Master List assembly can be overridden, for any reason, by providing this option. The *nnnn* must be replaced by a number in the range 4096-8182. This option is always accepted but only used if it occurs before the open of the CXX.

IXXNO=nnnn

The number of IXX buffers as set in the DBMSTLST Master List assembly can be overridden, for any reason, by providing this option. The *nnnn* must be replaced by a number in the range 3-9999. Some DBUTLTY functions need minimal data buffers and some would benefit with a large number. This option is always accepted but only used if it occurs before the first function that opens the CXX.

ON-ERROR-STOP

If an error condition exists, future functions (excluding SET and EDIT) are syntax edited but are not executed. This is the default at the start of each step.

Use of this function sets this action for future functions.

ON-ERROR-CONTINUE

If an error condition exists, future functions are edited and processed as though an error condition did not exist.

Note: Various system failures can occur that the operating system considers fatal and that prevent continued operation.

The error condition is not reset based upon the successful completion of this or a future function.

ON-ERROR-ABEND

Dictates action taken after the last function is complete for step. This is the default at the start of each step.

If an error condition then exists, the step is terminated with an z/OS ABEND 4 or a z/VSE CANCEL.

ON-ERROR-RC*n*

Dictates action taken after the last function is complete for step. The number in the parameter (*n*) can be 0, 4, 8, or 12.

If an error condition then exists, the step is terminated with the return code of the number provided.

ON-ERROR-REMOVE

Removes an error condition.

From this point following functions are edited and executed and, unless errors occur in them, the step terminate with a return code 0.

SNAPALL

This diagnostic tool is available, when requested by Technical Support, to help find a problem. Its purpose is to snap the address space after a utility function that is processed by a subordinate program completes, with or without an error condition. The reason this might be necessary is, for example, if a utility function's process, deeming itself successful and without error, completes but has not actually completed successfully after all.

SNAPERR

This diagnostic tool is available, when requested by Technical Support, to help find a problem. Its purpose is to snap the address space after a utility function, processed by a subordinate program, completes with an error condition. The reason this might be necessary is that some error conditions are expected to be obvious and no dump is taken, because a dump should not be necessary in order to correct the condition. If the error turns out not to be obvious, however, the SNAPERR can be used to help Technical Support with the additional information needed to find a solution.

Valid Entries:

OPTION1=ON-ERROR-STOP
OPTION1=ON-ERROR-CONTINUE
OPTION1=ON-ERROR-ABEND
OPTION1=ON-ERROR-RC*n*
OPTION1=ON-ERROR-REMOVE
OPTION1=SNAPALL
OPTION1=SNAPERR

Default Value:

OPTION1=ON-ERROR-STOP
OPTION1=ON-ERROR-ABEND

WTO--x

Specify WTO--*x* as shown, with WTO followed by two dashes, followed by a set of characters (replacing the *x* variable) that you want written to the console with a WTO in the DBUTLTY address space.

The set of characters specified by *x* have a length restriction imposed by the number of characters that fit on one input statement. Given that the SET OPTION1=WTO-- is 17 characters long and starts in column 1 (with the S in SET), and given that the last column in which the last possible character could occur is column 71, subtracting 17 from 71 gives a maximum of 54 user-characters (for the *x*), if blanks are not required (see following explanations) and 52 user-characters if blanks are required (due to two places being required for two single quote characters, as described in the next paragraphs).

If the set of characters that comprise the x have blank spaces embedded in them, they must be surrounded by single quotes. Otherwise, OPTION1= terminates on the first blank. If single quotes are used, however, those single quote marks are not printed. For example, BEGIN DATABASE 22 would be specified as follows:

```
SET OPTION1='WTO--BEGIN DATABASE 22'
```

An example of the result of the SET function just shown would be that it generates the message:

```
DB10099I - USER WTO--BEGIN DATABASE 22.
```

If the set of characters that comprises the x do not have blank spaces embedded in them, the single quotes are not needed. For example, BEGIN_DATABASE_22 would be specified as follows:

```
SET OPTION1=WTO--BEGIN_DATABASE_22
```

Note: If using a value for a keyword that starts with a quote, the end is the next quote. It is not possible to provide a value that has both a blank and a quote.

Possible uses for the WTO option include the following:

- The WTO option can be used to provide information about the progress of a DBUTLTY execution that has multiple functions. For example, if you were executing a series of functions for database 200, followed by a series of functions for database 300, before the database 200 executions begin, you could do the following:

  ```
  SET OPTION1=WTO--STARTING-DATABASE-2__
  ```

- Then, before the database 300 executions start, do the following:

  ```
  SET OPTION1=WTO--STARTING-DATABASE-3__
  ```

- You can use the WTO option to provide for a set of functions a meaningful descriptive name that would be useful to those who are watching the execution or reviewing the output. For example, you could specify the following:

  ```
  SET OPTION1='WTO--STARTING FUNCTIONS FOR PAYROLL DATABASE'
  ```

- If operator information is needed for some reason, the WTO option can be used to provide it.

Note: These actions could be done using a comment statement that would be imbedded in the print output but would not be in the console messages.

Example JCL

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...          See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
            SET OPTION1=ON-ERROR-STOP
            SET OPTION1=ON-ERROR-CONTINUE
 /*
 /&
 * $$ EOJ
```

Sample Report: SET

Following is a sample report showing the use of the SET function.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶               SET OPTION1=ON-ERROR-STOP

                    FUNCTION=SET
        ❷           OPTION1=ON-ERROR-STOP

        ❸
```

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
        ❶               SET OPTION1=ON-ERROR-CONTINUE

                    FUNCTION=SET
        ❷           OPTION1=ON-ERROR-CONTINUE

        ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

Chapter 34: SPILL and SPILLREW (Transfer Data to RXX)

Note: We recommend you use the new SPILLOPT function instead of the SPILL function. SPILL works the same as SPILLOPT SPILL=MAX (see SPILLOPT (Transfer Data to RXX Using MAX/MIN) (see page 715)).

SPILL or SPILLREW

The SPILL function transfers data from the Log Area (LXX) to the Recovery File (RXX). All spilling is controlled and done through the Multi-User Facility, which must be enabled to do a spill. The SPILL function is allowed for both active and inactive recovery. The SPILLREW function also transfers data, but provides the REWIND= parameter for you to specify rewinding instructions (see SPILLREW Function (see page 711)).

If blocks are spillable, the utility opens the Recovery File, copies the appropriate log blocks (from Multi-User) to the Recovery File, and closes the file. The space occupied by the spilled data is made available when the function is complete.

With an Active Recovery File, the spill is automatically done when the percentage is equal or above the LOGSPILL percentage and log blocks are spillable.

See the *CA Datacom/DB Database and System Administration Guide* for details about the LOGSPILL, LOGRCV, and MESSAGE Multi-User startup options.

Note: When using inactive recovery (that is, NO specified for the Multi-User startup option LOGRCV) and checking for spillable data based upon the LOGSPILL percentage, CA Datacom/DB presumes that the portion of the Log Area (LXX) that contains pipelined index and/or data need not be spilled when determining the number of blocks spillable. This allows a spill option to reduce index/data I/O when preparing for a spill.

When to Use Active Recovery

The SPILL function or SPILLREW function can be run at any time to request that the Multi-User Facility spill any spillable blocks to the Recovery File. This would be used in the situation of having run a job which ended but you need to run backward recovery to reverse the job. If the full job's logging records have not been spilled, run this spill. If any blocks are spillable (and a spill is not active), a spill is started. At its completion, the datetime stamp in the DB00310 message provides information to tell if the job is completely spilled. If not, wait a while and rerun the spill.

When to Use Inactive Recovery

With an Inactive Recovery File and when the console message DB00308 occurs, log blocks are spillable and a spill should be done. A spill may not be successful at other times for the following reasons:

- The Log Area may have data, but due to an active transaction, no blocks are spillable. The utility terminates processing the utility job step and takes no action concerning the Recovery File.
- A spill may already be in progress. In this case, a message so indicates and the utility terminates processing the utility job step and takes no action concerning the Recovery File.
- The Log Area might be empty. In this case, a message indicates the empty condition, and the utility continues to the next function without error and no action concerning the Recovery File.

Note: When using inactive recovery (that is, NO specified for the Multi-User startup option LOGRCV) and checking for spillable data based upon the LOGSPILL percentage, CA Datacom/DB presumes that the portion of the Log Area (LXX) that contains pipelined index and/or data need not be spilled when determining the number of blocks spillable. This allows a spill option to reduce index/data I/O when preparing for a spill.

SPILL Function

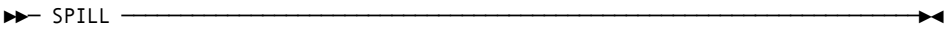
Multi-User Facility Connection

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

How to Use

Multi-User mode is required to execute this command. Use the following command format to transfer Log Area data to the Recovery File (RXX):



Command

SPILL

Invokes the function to transfer data from the Log Area (LXX) to the Recovery File (RXX).

Example JCL

The following shows the command to transfer data from the Log Area to the Recovery File.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and .
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // DLBL LXX,'lxx',,DA                              Log Area data set
 // EXTENT SYSnnn,volser
 // ASSGN SYSnnn,TAPE                              SYS number identified by LOGRSYS
 // PAUSE MOUNT TAPE
 // TLBL RXX,'rxx'                                  Recovery data set
 // EXEC DBUTLTY,SIZE=750K
          SPILL
 /*
 /&
 * $$ EOJ
```

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, JCL Requirements (see
page 34).
// ASSGN SYSnnn,DISK,VOL=volser,SHR
// DLBL CXX,'cxx',,DA                              Directory data set
// EXTENT SYSnnn,volser
// DLBL LXX,'lxx',,DA                              Log Area data set
// EXTENT SYSnnn,volser
// ASSGN SYSnnn,TAPE                              SYS number identified by LOGRSYS
// DLBL RXX,'rxx'                                  Recovery data set
// EXTENT,SYS000,volser,1,0,begtrk,ntrks
// ASSGN SYS000,DISK,VOL=volser,SHR
// EXEC DBUTLTY,SIZE=750K
          SPILL
 /*
/&
* $$ EOJ
```

Sample Report: SPILL

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶            SPILL

       ❷            FUNCTION=SPILL




       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


DB13085I - LOG AREA SPILL IS COMPLETE                                                          }
                                                                                               }
DB00310I - LOG SPILL 1-1, BLOCKS E-E4, TIMES ccyy/mm/dd hh.mm.ss-ccyy/mm/dd hh.mm.ss           }  ❹
                                                                                               }
DB00404I - RXX LAST RECORD DATE/TIME -  ccyy/mm/dd/ hh.mm.ss                                   }
```

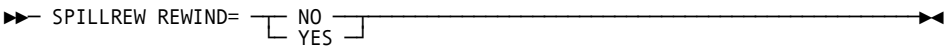
❹

Messages about the success of the spill. See the *CA Datacom/DB Message Reference Guide* for details on each message.

SPILLREW Function

How to Use

Multi-User mode is required to execute this command. Use the following command format to transfer Log Area data to the Recovery File (RXX) with rewind instructions:



Command

SPILLREW

Invokes the function to transfer data from the Log Area (LXX) to the Recovery File (RXX) with rewind instructions.

Required Keyword

REWIND=

YES indicates the tape is to be rewound before OPEN and after CLOSE. NO indicates no rewind before OPEN or after CLOSE. NO allows the stacking of spills on one tape.

Valid Entries:

NO or YES

Default Value:

(No default)

Example JCL

The following shows the command to transfer data from the Log Area to the Recovery File, with rewind instructions.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                            Directory data set
 // EXTENT SYSnnn,volser
 // DLBL LXX,'lxx',,DA                            Log Area data set
 // EXTENT SYSnnn,volser
 // ASSGN SYSnnn,TAPE                           SYS number identified by LOGRSYS
 // PAUSE MOUNT TAPE
 // TLBL RXX,'rxx'                                Recovery data set
 // EXEC DBUTLTY,SIZE=750K
         SPILLREW REWIND=NO
 /*
 /&
 * $$ EOJ
```

Sample Report: SPILLREW

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             SPILLREW

                    FUNCTION=SPILLREW
       ❷               REWIND=NO


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


DB13085I - LOG AREA SPILL IS COMPLETE                                                 }
                                                                                      }
DB00310I - LOG SPILL 1-1, BLOCKS E-E4, TIMES ccyy/mm/dd hh.mm.ss-ccyy/mm/dd hh.mm.ss } ❹
                                                                                      }
DB00404I - RXX LAST RECORD DATE/TIME - ccyy/mm/dd hh.mm.ss                            }
```

❹

Messages about the success of the spill. See the *CA Datacom/DB Message Reference Guide* for details on each message.

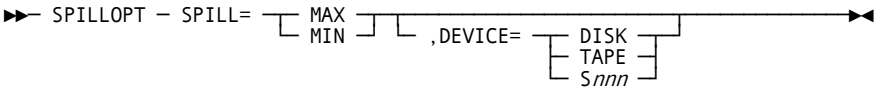
Chapter 35: SPILLOPT (Transfer Data to RXX Using MAX/MIN)

SPILLOPT

The SPILLOPT SPILL= function allows you to specify MAX or MIN when requesting a spill.

How to Use

Multi-User mode is required to execute this command. Use the following command format:



Command

SPILLOPT

Invokes the function to transfer data from the Log Area (LXX) to the Recovery File (RXX).

Required Keywords

SPILL=

The SPILLOPT SPILL= function allows you to specify MAX or MIN when requesting a spill.

Specifying SPILLOPT SPILL=MIN indicates that you want the spill to contain all that is spillable, *not including* the blocks controlling the pipelined index/data blocks. If performing a spill in the normal course of events, it is more efficient to use SPILL=MIN.

Specifying SPILLOPT SPILL=MAX indicates that you want the spill to contain the maximum number of blocks including pipeline index/data blocks. If performing a spill specifically to recover a job executed improperly, you should use SPILL=MAX in case the job recently finished.

Valid Entries:

MAX or MIN

Default Value:

(No default)

Optional Keyword

DEVICE=

The DEVICE= keyword specifies the device type for the output file.

When TAPE is being used, the system logical unit (SYS) number must be 001 (SYS001).

Use the S*nnn* entry to specify a specific SYS number (for tape only).

Valid Entries:

DISK, TAPE, S*nnn* (Certain values from previous versions are still accepted but are treated as DISK.)

Default Value:

(No default)

Example JCL

The following shows the command to transfer data from the Log Area to the Recovery File using SPILLOPT SPILL=MIN.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                          Directory data set
 // EXTENT SYSnnn,volser
 // DLBL LXX,'lxx',,DA                          Log Area data set
 // EXTENT SYSnnn,volser
 // ASSGN SYSnnn,TAPE                          SYS number identified by LOGRSYS
 // PAUSE MOUNT TAPE
 // TLBL RXX,'rxx'                              Recovery data set
 // EXEC DBUTLTY,SIZE=750K
          SPILLOPT SPILL=MIN
 /*
 /&
 * $$ EOJ
```

```
* $$ JOB ...           See the previous note and JCL Requirements (see page 34).
* $$ LST ...
// JOB name
// EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
// ASSGN SYSnnn,DISK,VOL=volser,SHR
// DLBL CXX,'cxx',,DA                          Directory data set
// EXTENT SYSnnn,volser
// DLBL LXX,'lxx',,DA                          Log Area data set
// EXTENT SYSnnn,volser
// DLBL RXX,'rxx'                              Recovery data set
// EXTENT,SYS000,volser,1,0,begtrk,ntrks
// ASSGN SYS000,DISK,VOL=volser,SHR
// EXEC DBUTLTY,SIZE=750K
          SPILLOPT SPILL=MAX
/*
/&
* $$ EOJ
```

Sample Report: SPILLOPT SPILL=MIN

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             SPILLOPT SPILL=MIN

       ❷             FUNCTION=SPILLOPT
                        SPILL=MIN


       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                               General Utility                                *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************


DB13085I - LOG AREA SPILL IS COMPLETE                                                          }
                                                                                               }
DB00310I - LOG SPILL 399-1, BLOCKS 31-61, TIMES ccyy/mm/dd hh.mm.ss-ccyy/mm/dd hh.mm.ss        }  ❹
                                                                                               }
DB00404I - RXX LAST RECORD DATE/TIME -  ccyy/mm/dd/ hh.mm.ss                                   }
```

❹

Messages about the success of the spill. See the *CA Datacom/DB Message Reference Guide* for details on each message.

Chapter 36: UNLOCK (Unlock a Data Area)

UNLOCK

The UNLOCK function allows you to reverse the LOCK function, therefore allowing any updates waiting to process to complete. See LOCK (Lock a Data Area) (see page 423) for information about this function.

When to Use

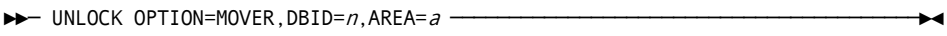
Use the UNLOCK function to reverse a LOCK function.

This function communicates with one Multi-User Facility. Be aware of the following:

- Ensure that DBUTLTY is communicating with the desired Multi-User Facility by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the Multi-User Facility, and
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

How to Use

Multi-User mode is required to execute this command. Use the following command format to issue an UNLOCK function:



Command

UNLOCK

Invokes the function to reverse the LOCK function, therefore allowing any updates waiting to process to complete.

Required Keywords

OPTION=MOVER

Requests the option that inhibits movement of data records due to an update of a compressed data area.

DBID=

Identifies the database you want to unlock.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

AREA=

Identifies the area you want to unlock.

Valid Entries:

DATACOM-NAME of the area in the database specified

Default Value:

(No default)

Example JCL

The following shows the command to unlock a data area.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
      UNLOCK OPTION=MOVER,DBID=1,AREA=DEM
 /*
 /&
 * $$ EOJ
```

Sample Report: UNLOCK

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
       ❶             UNLOCK OPTION=MOVER,DBID=1,AREA=DEM

                    FUNCTION=UNLOCK
       ❷               OPTION=MOVER
                       DBID=1
                       AREA=DEM



       ❸
```

❶

The command exactly as entered.

❷

An analysis of keywords encountered and expected. Any errors found are flagged with a note in the left margin.

❸

Any messages related to syntax processing.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     2
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

                     *                                     REQUEST COMPLETE     ❹
```

❹

Message indicating that the UNLOCK completed successfully.

Chapter 37: VERINDEX (Verify Index)

VERINDEX (Verify Index)

The VERINDEX (Verify Index) function is used to ensure the integrity of the CA Datacom data rows and their corresponding keys in the index.

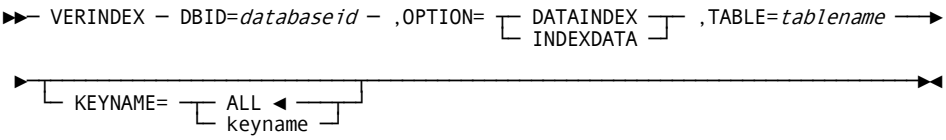
When to Use

This function is designed to be used when you question the integrity of the index with the data, but the table cannot be taken offline. VERINDEX executes while the table is open for processing.

The function is intended to pass through the data and index areas producing a list of the index and data mismatches. You can review the output listing to make a determination of the action to be taken next. If there is only one index and data mismatch, you can attempt to fix it manually by deleting and adding a row again rather than rebuilding the entire index.

How to Use

Use the following format for this function:



Command

VERINDEX

Invokes the function to verify the integrity of the index and data.

Required Keywords

DBID=

Specifies the ID of the database.

Valid Entries:

database ID

Default Value:

(No default)

,OPTION=

Specifies the Option entry.

Valid Entries:

DATAINDEX or INDEXDATA

Default Value:

(No default)

,TABLE=

Specifies the three-character name of the table.

Valid Entries:

table name

Default Value:

(No default)

Optional Keyword

KEYNAME=

Specifies the five-character name of the key in the table, if only one key in the table is to be verified. Omit the keyword to process all keys for the table.

Valid Entries:

key name

Default Value:

ALL

Data to Index (DATAINDEX)

The DATAINDEX option processes each data row for updating using physical data row order GSETP/GETPS commands with row locking. Once a data row is retrieved, the key definition information is used to construct the key value that should be in the index. The LOCKI (locate key intersection) command is issued to determine if there is an index entry with this key value that points to the data row. The process is repeated for each of the key definitions.

If an index entry is missing and an error message is produced, the processing then continues to the next key definition or data row. For any error, a flag is set so that at the end of the function execution a condition code is recognized.

The error message provides information about the record found and the key value that should exist but does not.

If an unexpected error occurs after a request, a system or program failure has occurred and the function terminates normally to DBUTLTY.

Once all data rows and their index entries have been processed, a final status message is produced which includes the data row counts, index entry counts, index error counts, and total I/O counts.

The process locks the data rows one at a time. The lock lasts until the index verifications are done. After the indexes are verified, the data row is unlocked and processing moves on to the next data row.

During this process, no errors are expected. If other tasks are performing maintenance concurrent with this process, it is possible for another task to UPDAT a record that is compressed in a manner that could move it from a block to the right of the task of the current physical block being processed to the left of the current block. In this event, it is possible for the data to index process to overlook this record and not verify its index values. This occurrence, however, is very rare.

A key can be defined in CA Datacom/DB with the attribute INCLUDE-NIL-KEY that can be set to NO, where key values that are completely blanks or completely binary zeros are excluded from the index and are therefore not available for checking. This means that a key with INCLUDE-NIL-KEY that is all blanks or binary zeros is not checked or counted. This option is not available for the native sequence key nor the Master Key. This condition creates the possibility that counts of various keys might be different.

Example JCL

The following shows sample JCL for DATAINDEX.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and JCL Requirements (see page 34).
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see
page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
       VERINDEX DBID=001,OPTION=DATAINDEX,TABLE=PAY
       VERINDEX DBID=001,OPTION=INDEXDATA,TABLE=PAY
 /*
 /&
 * $$ EOJ
```

Sample Report: DATAINDEX without an Error

The following is a sample report of the DATAINDEX function without an error.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                              CA Datacom/DB                                   *
Time: hh.mm.ss      *                            General Utility                                   *      Release: 12
                    *                 Copyright © 2009 CA. All rights reserved.                    *              SPnn
                    ********************************************************************************

 KEY RELATED STATISTICS:
 KEYID   KEYNAME   MAST   MATCH_COUNT   ERROR_COUNT
     1     TST01      Y         21999             0
     2     TST02      N         21999             0
     3     TST03      N         21999             0
     4     TST04      N         21999             0
     5     TST05      N         21999             0
     6     TST06      N         21999             0
     7     TST07      N         21999             0
     8     TST08      N         21999             0
     9     TST09      N         21999             0
 FINAL STATISTICS:
   INDEX MATCHES        197991
   INDEX ERRORS              0
   I/O OPERATIONS        64524
   ROWS PROCESSED        21999
```

Sample Report: DATAINDEX with an Error

The following is a sample report of the DATAINDEX function with an error.

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************
 KEY TST02 NO INDEX FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 00004478 DATA TTTR 00000106
 00000000  C2875E07 BE0F7C83                                                          *Bg
...@c
```

Index to Data (INDEXDATA)

The INDEXDATA option processes each key definition selected for the table. For each key definition, all index values are processed.

The index uses locate commands, LOCKG/LOCNX for processing. For each index entry, the matching data row is retrieved using a REDLE command. REDLE commands offer no exclusive control or protection from other concurrent tasks performing maintenance to the same table. If other tasks are not blocked from maintenance, many false errors can occur and must be handled or verified in some other way outside the scope of this DBUTLTY function. A data row not found is considered in doubt and an error message is generated. The processing then continues to the next index entry.

An unexpected return code indicates a system or program failure and causes the DBUTLTY function to terminate with the failure.

If a doubtful index condition occurs, a flag is set and at the end of the function, a normal DBUTLTY error is set.

Once the key definitions, their index entries, and their matching data rows have been processed, a final report is printed with details of the function.

The function does not lock the data rows. When an attempt is made to retrieve the data row and a RC=04 or RC=12 occurs, this means that the data row is not present at the time but could have been present when the index entry was retrieved. Errors in this case are considered in doubt.

To be sure the index and data are stable and that errors are real, perform the ACCESS=NOMAINT function on the DBID. This prevents any deletions and protects the LOCKG/REDLE processing. If the table must remain open to other users, do not restrict access and simply run the VERINDEX function. If errors are detected, run the same key name a second time. Key values that are errors on both runs would likely be true errors.

A key can be defined in CA Datacom/DB with the attribute INCLUDE-NIL-KEY that can be set to NO, where key values that are completely blanks or completely binary zeros are excluded from the index and are therefore not available for checking. This means that a key with INCLUDE-NIL-KEY that is all blanks or binary zeros is not in the index and therefore cannot be used for comparison against the data. This option is not available for the native sequence key nor the Master Key. This condition creates the possibility that counts of various keys might be different.

Example JCL

The following shows sample JCL for INDEXDATA.

Note: Use the following as a guide to prepare your JCL. The JCL statements are for example only. Lowercase letters in a statement indicate a value you must supply. Code all statements to your site and installation standards.

```
 * $$ JOB ...           See the previous note and .
 * $$ LST ...
 // JOB name
 // EXEC PROC=procname  Whether you use PROCs or LIBDEFs, see JCL Requirements (see page 34).
 // ASSGN SYSnnn,device-type,VOL=volser,SHR
 // DLBL CXX,'cxx',,DA                              Directory data set
 // EXTENT SYSnnn,volser
 // EXEC DBUTLTY,SIZE=750K
       VERINDEX DBID=2,OPTION=INDEXDATA,TABLE=AGR
 /*
 /&
 * $$ EOJ
```

Sample Report: INDEXDATA

The following is a sample report of the INDEXDATA function.

```
Date: mm/dd/ccyy    ********************************************************************************          Page:     1
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Release: 12
                    *                  Copyright © 2009 CA. All rights reserved.                   *               SPnn
                    ********************************************************************************

 FUNCTION=VERINDEX
                       DBID=00997
                       OPTION=INDEXDATA
                       TABLE=K01

 KEY RELATED STATISTICS:
  ❶        ❷        ❸          ❹             ❺
 KEYID   KEYNAME   MAST   MATCH_COUNT   ERROR_COUNT
     1     TST01      Y         21999             1
     2     TST02      N         21999             1
     3     TST03      N         21999             1
     4     TST04      N         21999             1
     5     TST05      N         21999             1
     6     TST06      N         21999             1
     7     TST07      N         21999             1
     8     TST08      N         21999             1
     9     TST09      N         21999             1
 FINAL STATISTICS:
   INDEX MATCHES        197991
   INDEX ERRORS              9
   I/O OPERATIONS        65688
                    DB10045E - ABNORMAL TERMINATION DUE TO ERROR(S)
```

❶

KEYID displays the identification number of key.

❷

KEYNAME displays the name of key.

❸

MAST identifies whether the key is a Master Key. Set to Yes or No.

❹

MATCH_COUNT displays the count of index entry key values found that matched data record key values.

❺

ERROR_COUNT displays the count of the index entries that are in doubt. For each of the in doubt index values, a message and a dump of the key value are written to SYSLST. The following section displays an example of this output.

Example Dump: INDOUBT Index Entries

For each of the index values that are in doubt, a message and a dump of the key value are provided in the SYSOUT data set INDOUBT.

```
KEY TST01 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  F0F5F4F2 F9F3F9F6                                        *05429396                       *
KEY TST02 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  C28EBF77 49D8F401                                        *B....Q4.
KEY TST03 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  F8F0F5F3 F6F2F8F8                                        *80536288                       *
KEY TST04 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  F0F0F0F0 F7F5F7F5                                        *00007575                       *
KEY TST05 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  C28EBF77 49D8F401                                        *B....Q4.                       *
KEY TST06 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  00000000 00000000                                        *........                       *
KEY TST07 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  40404040 40404040 40404040 40404040   40404040 40404040                                 *                               *
KEY TST08 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  F9F9F9F9 F2F4F2F4                                        *99992424                       *
KEY TST09 NO RECORD FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103
00000000  F2F4F3F3 F6F6F1F4 F8F0F5F3 F6F2F8F8   F0F5F4F2 F9F3F9F6 F0F0F0F0 F7F5F7F5   *2433661480536288054293960000757 5*
00000020  C28EBF77 49D8F401 00000000 00000000   F9F9F9F9 F2F4F2F4 40404040 40404040   *B....Q4.........99992424        *
00000040  40404040 40404040 40404040 40404040   D2F0F140                            *               K01             *
```

The message provides the key name being processed with the URI value and the data block pointer in TTTR (track and record on track) format. The format is like a SYSUDUMP with the memory address in the first 8 positions, the hexadecimal formatted value, and the translation on the right.

Index

A

ACCESS function • 46, 48, 55
  area level ACCESS • 48
  area level DBUTLTY control • 46
  database ACCESS • 46
  description • 55
Accounting Facility • 65, 66, 72, 75, 81, 85, 144
  activating • 65
  closing tables • 72
  compiling table definitions • 66
  deactivating • 72
  forcing data to disk • 85
  opening tables • 81
  Parameter Table (PRM) • 66
  summary information produced at COMM EOJ • 144
  validating table definitions • 75
Accounting Facility Summary Information • 144
ACCT function • 65, 66, 71, 75, 81, 85
  CATALOG option • 66, 71
  CLOSE option • 71, 75
  EDIT option • 75, 81
  OPEN option • 81, 85
  SPILL option • 85
ALTER option (COMM function) • 116, 124
ALTER option (CXXMAINT function) • 206, 209, 212, 215, 218, 221, 225, 228, 235, 238
  CBSUSE • 206, 209
  DBCS • 209, 212
  DSN • 212, 215
  DSOP • 215, 218
  LANGUAGE • 218, 221
  LINK • 221, 225
  LOGGING • 225, 228
  SINGLE • 235, 238
  Single User mode • 235
Area level DBUTLTY control • 43, 46, 49, 50, 56, 124, 126
  ACCESS function • 46, 56
  ACCESS Multi-User startup option • 50
  COMM CLOSE function • 49, 124, 126
  EXTEND function • 43
  INIT function • 43
  LOAD function • 43
  overview • 43
  REORG function • 43
  restrictions and requirements • 50
  RETIX function • 43
  table MUF_ACCESS_AREA (MFN) • 49
AREA= keyword • 101, 317, 323, 354, 371, 376, 378, 423, 472, 484, 502, 530, 643, 666, 670, 689, 719
  BACKUP function • 101
  EXTEND function • 317
  EXTRACT function • 323
  LOCK function • 423
  REMOVE function • 472
  REORG function • 484
  RETIX function • 689
  UNLOCK function • 719
Arithmetic symbols in syntax diagrams • 26
Automatic DBUTLTY STATUS • 37

B

Backup files • 305
  extracting data from • 305
BACKUP function • 89, 95, 96, 101, 110, 112
  CXX option • 89, 95
  data area option • 95
  data area option while processing data • 96
  data area option, calculating block size • 101
  Native Key sequence • 110
  parallel backup • 110
  physical sequence • 112
Backups (concatenations) • 335
  using FILEIN function • 335
BLKSIZE= keyword • 101, 323, 364, 371, 376, 430, 484
  BACKUP data area • 101
  EXTRACT function • 323
  INIT FXX function • 376
  INIT IXX function • 364
  INIT LXX function • 371
  MASSADD function • 430
  REORG function • 484
BLOCKS= keyword • 376
  INIT FXX function) • 376
BLOCKS= keyword (INIT LXX function) • 371
BUFFERS= keyword (COMM ALTER function) • 116

C

CA Datacom DB2 Transparency indicated on Statistics and Diagnostics Area (PXX) report • 568
CA Ideal Utilities • 25
CA IPC Utilities • 25
CATALOG option (ACCT function) • 66, 71
CBSUSE= keyword (CXXMAINT ALTER function) • 206, 209
CHECK PENDING status for tables • 199
CLOSE option • 71, 73, 75, 124, 125, 129
  ACCT function • 71, 75
  COMM function • 124, 129
  console command • 73, 125
CLRPXX option • 129, 130, 133
  COMM function • 129, 133
  console command • 130
CMPRS= keyword (BACKUP data area function) • 101
COMM • 175
  STATS • 175
COMM function • 116, 124, 125, 129, 130, 133, 135, 136, 162, 163, 166, 167, 171, 175, 176, 179, 180
  ALTER option • 116, 124
  CLOSE console command • 125
  CLOSE option • 124, 129
  CLRPXX console command • 130
  CLRPXX option • 129, 133
  CONSOLE option • 133, 135
  EOJ option • 136, 162
  EOJ, EOJFREE, EOJKEEP options • 162
  flushing pipeline (COMM STATS) • 176
  NEWRXX console command • 163
  NEWRXX option • 162, 166
  REQABORT console command • 167
  REQABORT option • 166, 171
  SNAP console command • 171
  SNAP option • 171, 175
  STATS console command • 176
  STATS option • 175, 179
  STATUS console command • 180
  STATUS option • 180
Comma • 29
  repeat symbol, use in • 29
Commands • 33
  multiple in one job stream • 33
COMPER= keyword (REPORT DEVICE function) • 650
Compound Boolean Selection Facility • 206, 594, 606
  ALTER CBSUSE option, CXXMAINT function • 206
  Diagnostic Report • 594, 606
  statistical population counting on Statistics and Diagnostics Area (PXX) Report • 606
COMPRESS= keyword (REPORT DEVICE function) • 650
Compressed data • 98, 423
  backing up • 98
  preventing movement of compressed records • 423
Concatenating backups • 335
  using FILEIN function • 335
CONFIRM function • 199
Console commands • 37, 175
  COMM STATS • 175
  STATUS (DBUTLTY) • 37
CONSOLE option • 133, 135
  COMM function • 133, 135
CONSOLE= (DBSYSID macro parameter) • 37
CONSOLE= parameter (DBSYSID) • 37
Constraints, verifying integrity • 199
Continuous operation, BACKUP function • 96
CONVERT1112 option (CXXMAINT function) • 238
CONVERT1211 option (CXXMAINT function) • 246
CXX • 675
  update • 675
CXX option • 89, 95, 353, 357, 386, 393, 501, 529, 665, 669
  BACKUP function • 89, 95
  INIT function • 353, 357
  LOAD function • 386, 393
  REPORT function • 501, 529
  RESET function • 665, 669
CXXMAINT function • 206, 209, 212, 215, 218, 221, 225, 228, 235, 238, 246, 253, 264
  ALTER CBSUSE option • 206, 209
  ALTER DBCS option • 209, 212
  ALTER DSN option • 212, 215
  ALTER DSOP option • 215, 218
  ALTER LANGUAGE option • 218, 221
  ALTER LINK option • 221, 225
  ALTER LOGGING option • 225, 228
  ALTER SINGLE option • 235, 238
  ALTER Single User Mode • 235
  CONVERT1112 option • 238
  CONVERT1211 option • 246
  DDPROD option • 246, 253
  DELETE option • 253
  FORMAT0201 option • 264
  PURGE option • 264
CXXNAME= keyword (INIT CXX function) • 354

CYCLE= keyword • 484
REORG function • 484
CYCLE= keyword (RXXFIX function) • 691

D

Data area option • 95, 96, 110, 357, 362, 393
BACKUP function • 95
BACKUP function while processing data • 96
INIT function • 357, 362
LOAD function • 393
Native Key sequence • 110
Data areas • 97, 212, 215, 218, 315, 357, 362, 381, 395, 396, 399, 423, 649, 656, 723
changing data set name • 212, 215
changing space management option • 215, 218
extending • 315
initializing • 357, 362
linking to Directory • 381
loading with multiple tables • 399
null loading • 395
preventing movement of compressed records • 423
reorganization impact on recovery • 97
reorganizing • 396
space estimation report • 649, 656
verifying with VERINDEX function • 723
DATABASE= keyword • 67, 76
ACCT CATALOG function • 67
ACCT EDIT function • 76
Databases • 249, 253, 394
deleting definitions • 253
loading • 394
NEWDBID= keyword, CXXMAINT DDPROD function • 249
DATACOM= keyword (INIT CXX function) • 354
DATAINDEX option, VERINDEX function • 725
DATE= keyword (RECOVERY function) • 451
DBCS= keyword (CXXMAINT ALTER function) • 209, 212
DBID= keyword • 101, 254, 259, 262, 295, 317, 323, 364, 382, 423, 430, 437, 450, 472, 484, 494, 502, 530, 550, 659, 666, 689, 691, 719
BACKUP data area function • 101
CXXMAINT function DELETE option • 254
CXXMAINT function FORMAT0102 option • 259, 262
DEFRAG function • 295
EXTEND function • 317
EXTRACT function • 323
LINK function • 382
LOCK function • 423
MASSADD function • 430
OLREORG function • 437
RECOVERY function • 450
REFGROUP report • 550
REMOVE function • 472
REORG function • 484
REPLACE function • 494
RESET function, CXX option • 666
RETIX function • 689
RXXFIX function • 691
UNLOCK function • 719
DBID= keyword (RXXFIX function) • 691
DBIDCPR program • 269, 283, 286
DBIDPPR program • 269, 286
DBSYSID macro • 58, 116, 163, 168, 172, 177, 181
DELAY68= • 58, 116
parameters • 163, 168, 172, 177, 181
DELAY68= • 163, 168, 172, 177, 181
DBSYSID macro parameter (DBSIDPR module) • 37
CONSOLE= parameter • 37
DBTEST function • 269, 270, 272, 275, 276, 277, 280, 283, 286
control statements • 277, 280
DBTEST command • 276
DBUREND command • 275
DBURSTR command • 270
DBURTBL command • 272
Index Analysis Program (DBIDCPR) • 283, 286
Index Population Analysis Program (DBIDPPR) • 286
DBTSTPR (CA Datacom/DB Test Program) • 277
DBTSTPR program • 269
DBUTLTY • 175
console commands • 175
DDOVER= keyword (ACCT EDIT function) • 76
DDPROD option (CXXMAINT function) • 246, 253
Default values in syntax diagrams • 30
DEFRAG function • 293
DELAY68= (DBSYSID keyword) • 58, 116, 126, 131, 133, 137, 163, 168, 172, 177, 181
DELETE option (CXXMAINT function) • 253
Delimiters • 27
syntax diagrams, use in • 27
DEVICE= keyword • 101, 249, 323, 335, 341, 430, 484, 494
BACKUP function • 101

CXXMAINT DDPROD function • 249
EXTRACT function • 323
FILEIN function • 335
FILEOUT function • 341
MASSADD function • 430
REORG function • 484
REPLACE function • 494
DEVICE= keyword (REPORT function) • 649, 656
Diagnostic Information Report (miscellaneous information) • 147
Directory (CXX) • 205, 238, 246, 253, 264, 353, 354, 357, 386, 393, 501, 529, 665, 669
base level full Version 12.0 features • 258, 261
conversion from previous version • 238
deleting definitions • 253
downgrading from prior version • 246
format • 354
initializing • 353, 357
loading • 386, 393
maintaining • 205
reporting • 501, 529
RESET function • 665, 669
updating definitions • 246, 253
DSN= keyword (CXXMAINT ALTER function) • 212, 215
DSOP= keyword (CXXMAINT ALTER function) • 215, 218
DTFNAME= keyword • 101, 249, 323, 341, 430, 484
BACKUP function • 101
CXXMAINT DDPROD function • 249
EXTRACT function • 323
FILEOUT function • 341
MASSADD function • 430
REORG function • 484
DUMPS= keyword (COMM ALTER function) • 116

E

EDIT function • 301
EDIT option (ACCT function) • 75, 81
End-of-file mark, recovering on Recovery File • 691
EOJ option (COMM function) • 136, 162
EOJ, EOJFREE, EOJKEEP options (COMM function) • 162
ETIME= keyword (RECOVERY function) • 451
EXCLUDE= keyword (BACKUP data area function) • 101
Executing multiple commands in one job stream • 33
Expression tree dump on Statistics and Diagnostics Area (PXX) Report • 590
EXTBKUP function • 305
EXTEND function • 315
EXTRACT function • 321, 331
parallel extract • 331

F

FILEIN function • 335
FILEOUT function • 341
FIRSTKEY= keyword • 101, 295, 323, 437, 550
BACKUP data area function • 101
DEFRAG function • 295
EXTRACT function • 323
OLREORG function • 437
REFGROUP report • 550
FLEXPOOL function • 347
console command • 347
Flushing pipeline • 176
Force Area (FXX) • 375
initializing • 375
FORMAT0201 option (CXXMAINT function) • 264
Functions, multiple in one job stream • 33
FXX option • 375
INIT function • 375

G

GROWTH= keyword (REPORT DEVICE function) • 650

H

History Report (Type H) • 659
HISTORY= keyword • 691
RXXFIX function • 691

I

IGN68= keyword (ALTER function) • 58
IGN68= keyword (COMM function) • 116, 126, 132, 135, 139, 165, 169, 173, 179, 183
Impact Report (TYPE=I) • 247, 527, 529
INCLUDE= keyword (BACKUP data area function) • 101
Index Area (IXX) • 212, 215, 283, 286, 293, 315, 347, 362, 364, 369, 381, 471, 529, 545, 673, 688, 723
block size, changing • 364
buffers with FLEXPOOL function • 347
changing data set name • 212, 215
defragmenting • 293
extending • 315

Index Analysis (DBIDCPR) • 283, 286
Index Population Analysis (DBIDPPR) • 286
initializing • 362, 369
linking to Directory • 381
rebuilding • 673, 688
removing a key • 471
reporting • 529, 545
reviewing distribution of key values • 286
reviewing for correctness • 283, 286
verifying with VERINDEX function • 723
INDEXDATA option, VERINDEX function • 728
INIT function • 353, 357, 362, 369, 375
CXX option • 353, 357
data area option • 357, 362
FXX option • 375
IXX option • 362, 369
LXX option • 369
INNUM= keyword • 484
REORG function • 484
Input data sets • 335
defining with FILEOUT • 335
IOERROR= keyword (BACKUP data area function) • 101
IXX option • 362, 369, 529, 532, 535, 538, 541, 545, 673
INIT function • 362, 369
rebuilding data space index • 673
REPORT function • 529, 532, 535, 538, 541, 545

J

JCL • 34
requirements • 34
JCL requirements • 34
JOBNAME= keyword • 116, 451
COMM ALTER function • 116
RECOVERY function • 451

K

KEY entity-occurrences • 471, 535, 541
changing length • 471
Index Area report sequenced by key ID • 535
removing from Index Area • 471
reporting unique key values • 541
KEYID= keyword • 295, 472, 530
DEFRAG function • 295
REMOVE function • 472
REPORT function - IXX • 530
KEYNAME= keyword (RETIX function) • 689
Keys • 206, 209, 286
trial deletion of • 206, 209
value distribution analysis • 286
Keywords in syntax diagrams • 26

L

LABEL= keyword • 101, 323, 335, 341, 430, 484, 494
BACKUP function • 101
EXTRACT function • 323
FILEIN function • 335
FILEOUT function • 341
MASSADD function • 430
REORG function • 484
REPLACE function • 494
LANGUAGE= keyword (CXXMAINT ALTER function) • 218, 221
LASTKEY= keyword • 101, 295, 323, 437, 550
BACKUP data area function • 101
DEFRAG function • 295
EXTRACT function • 323
OLREORG function • 437
REFGROUP report • 550
Libraries • 34
Libraries for CA Datacom products • 34
LINK function • 381
LINK= keyword (CXXMAINT ALTER function) • 221, 225
LOAD function • 385, 386, 393, 394
CXX option • 386, 393
data area option • 393
loading a database • 394
overview • 385
Loading (backups) • 335
defining with FILEIN • 335
LOADPTN= keyword • 41
LOCK function • 423
Log Area (LXX) • 369, 370, 545, 549, 669, 707, 715
full • 707
initializing • 369
reporting • 545, 549
RESET function • 669
spilling • 707, 715
variable log block size • 370
LOGGING= keyword (CXXMAINT ALTER function) • 225, 228
LXX option • 369, 545, 549, 669
INIT function • 369
REPORT function • 545, 549

RESET function • 669

M

MASSADD function • 427
Master List dump example • 175
Master List, printing a dump • 171
MASTERL= keyword (COMM ALTER function) • 116
Memory Resident Data Facility (MRDF) • 142
summary information produced at COMM EOJ • 142
Memory Resident Data Facility Summary Information report • 142
Memory Use Summary Information • 154
Messages • 34
Miscellaneous Diagnostic Information Report • 147
MOVER option • 423, 719
LOCK function • 423
UNLOCK function • 719
MUF EOJ Log Status Information • 158
Multiple commands in one job stream • 33
Multi-User Facility • 136, 141, 166, 171, 180, 563
aborting a request • 166, 171
ending processing • 136
Multi-User startup option list produced at COMM EOJ • 141
PXXSTATS • 563
status of jobs • 180
MULTUSE= keyword • 43, 101, 270, 323, 430, 494
BACKUP function • 101
DBTEST function • 270
EXTEND function • 43
EXTRACT function • 323
INIT function • 43
LOAD function • 43
MASSADD function • 430
RECOVERY function (forward) • 43
REORG function • 43
REPLACE function • 494
RETIX function • 43

N

Native sequence extract • 321
NEWDBID= keyword • 249
CXXMAINT DDPROD function • 249
NEWRXX console command • 163
NEWRXX option (COMM function) • 162, 166
NODE= keyword (RECOVERY function) • 451
NUMBER= keyword • 437
OLREORG function • 437

O

OLDEP= keyword • 364
OPEN option • 81, 82, 85
ACCT function • 81, 85
console command • 82
OPTION= keyword • 259, 262
CXXMAINT function FORMAT0102 option • 259
CXXMAINT function FORMAT0201 option • 262
OPTION1= keyword • 494
REPLACE function • 494
OPTION2= keyword • 484
REORG function • 484
OPTION2= keyword (CXXMAINT ALTER function) • 228, 235
ORIGRUNU= keyword (RECOVERY function) • 451
Output data sets • 341
defining with FILEOUT • 341

P

Parameter Table (PRM) (Accounting Facility) • 66
Parentheses • 27
syntax diagrams, use in • 27
PARM= keyword (COMM ALTER function) • 116
Partitioned tables • 40, 41, 42
BACKUP DBUTLTY function • 40
EXTRACT DBUTLTY function • 41
LOAD DBUTLTY function • 41
LOADPTN= keyword • 41
RECOVERY DBUTLTY function • 42
REORG DBUTLTY function • 42
PERCENT= keyword • 295
DEFRAG function • 295
Physical sequence extract • 321
Pipeline, use by MASSADD • 427
Procedure Summary Information • 152
Procedure TCB Use Summary Information • 151
Programs • 26, 27, 29
comma • 29
repeat symbol, use in • 29
parentheses • 27
syntax diagrams, use in • 27
punctuation • 26
syntax diagrams, use in • 26
Punctuation marks in syntax diagrams • 26
PURGE option (CXXMAINT function) • 264
PXX option (REPORT function) • 562

PXXSTATS Multi-User startup option • 563
impact on Statistics and Diagnostics Area (PXX) Report • 563

R

RCERROR= keyword (COMM ALTER function) • 116
Rebuilding the Recovery File • 691
RECBASE command • 450
RECID keyword • 444
RECID= keyword (BACKUP data area function) • 101
RECJOB command • 451
Records • 675
recount • 675
RECORDS= keyword (REPORT DEVICE function) • 650
Recount records • 675
Recovery File (RXX) • 162, 166, 642, 649, 691, 707, 715
active • 707
Analysis Report • 642
inactive • 707
NEWRXX option (COMM function) • 162, 166
recovering end-of-file mark • 691
requesting new file • 162, 166
spilling Log Area (LXX) • 707, 715
Split File Analysis Report • 642, 649
RECOVERY function • 97, 443, 444, 450, 451
backups for forward recovery • 97
data area reorganization • 97
RECBASE command • 450
RECID keyword • 444
RECJOB command • 451
RECOVERY= keyword (REPORT DEVICE function) • 650
RECSIZE= keyword (REPORT DEVICE function) • 650
REFGROUP • 549, 562
reporting • 549, 562
REFGROUP option (REPORT function) • 549, 562
REFGROUP= keyword • 437, 550
OLREORG function • 437
REFGROUP report • 550
Relative branching support • 549
REMOVE function • 471
REORG function • 475, 484
AREA= keyword • 484
BLKSIZE= keyword • 484
CYCLE= keyword • 484
DBID= keyword • 484
DEVICE= keyword • 484
DTFNAME= keyword • 484
INNUM= keyword • 484
LABEL= keyword • 484
OPTION2= keyword • 484
REWIND= keyword • 484
SEQBUFS= keyword • 484
REPLY= keyword (RESET CXX function) • 666
Report function • 247, 501, 527, 529, 532, 535, 538, 541, 545, 549, 562, 590, 594, 606, 629, 649, 656, 659
Compound Boolean Selection Facility diagnostic report • 594
CXX (Directory) • 501, 529
DEVICE (data area space estimation) • 649, 656
History (Type H) • 659
Impact Report (TYPE=I) • 247
IXX (Index Area) • 529, 532, 535, 538, 541, 545
LXX (Log Areas) • 545, 549
PXX (Statistics and Diagnostics Area) • 562, 590
REFGROUP • 549, 562
RXX (Recovery File) • 629, 649
statistical population counting • 606
TYPE=I (Impact Report) • 247, 527
Utility Function History (Type H) • 659
Reports • 141, 142, 144, 147, 151, 152, 153, 154, 158, 160, 175, 247, 527, 659
Accounting Facility Summary Information • 144
History (Type H) • 659
Impact Report (TYPE=I) • 247, 527
Master List dump example • 175
Memory Resident Data Facility Summary Information • 142
Memory Use Summary Information • 154
Miscellaneous Diagnostic Information report • 147
MUF EOJ Log Status Information • 158
Multi-User Startup Options Echoed • 141
other information • 160
Procedure Summary Information • 152
Procedure TCB Use Summary Information • 151
Return Code Summary Information • 153
SQL Code Summary Information • 153
Startup Options Echoed report • 141
task information • 160
TYPE=I (Impact Report) • 247, 527
Utility Function History (Type H) • 659
REQABORT console command • 167
REQABORT option (COMM function) • 166, 171

REQSEQNO= keyword (COMM REQABORT function) • 168
RESET function • 665, 669
  CXX option (Directory) • 665, 669
  LXX option (Log Areas) • 669
RESET option (SECURITY function) • 695
RETIX function • 673, 675, 688
  Data space index • 673
  KEYNAME=*SETR (recounting records) • 675
  recount records (KEYNAME=*SETR) • 675
Return Code Summary Information • 153
REWIND= keyword • 101, 323, 335, 341, 430, 484, 494, 711
  BACKUP function • 101
  EXTRACT function • 323
  FILEIN function • 335
  FILEOUT function • 341
  MASSADD function • 430
  REORG function • 484
  REPLACE function • 494
  SPILLREW function • 711
RUNUNIT= keyword (RECOVERY function) • 451
RXX option (REPORT function) • 629, 649
RXXFIX function • 691

S

Security • 33
  considerations for DBUTLTY • 33
Security considerations • 295, 436, 549
  DEFRAG function • 295
  OLREORG function • 436
  REFGROUP report • 549
SECURITY function • 695
SEQ= keyword • 101, 323
  BACKUP data area function • 101
  EXTRACT function • 323
SEQBUFS= keyword • 99, 101, 272, 323, 484
  BACKUP data area function • 99, 101
  DBURTBL function • 272
  EXTRACT function • 323
  REORG function • 484
SID (sub-identifier ID) • 181
Simulate console command • 133
Single User mode (CXXMAINT ALTER) • 235
Single User security • 235, 238
SINGLE= keyword (CXXMAINT ALTER function) • 235, 238
SLACK= keyword • 323, 494
  EXTRACT function • 323
  REPLACE function • 494
SNAP console command • 171
SNAP option (COMM function) • 171, 175
SORT= keyword • 494
  REPLACE function • 494
SORTWK= keyword • 494
  REPLACE function • 494
Space estimation report • 649, 656
Space management options for data areas • 215, 218
Space Utilization Report • 523, 527
SPILL console command for Accounting Facility • 85
SPILL function • 707, 711
SPILL option (ACCT function) • 85
SPILLOPT function • 715
SPILLREW function • 711
SQL Code Summary Information • 153
SQL dumps • 608
SQL processing • 99, 116, 254, 590
  backing up data • 99
  deleting definitions from the CXX • 254
  expression tree dump on Statistics and Diagnostics Area (PXX) Report • 590
  tracing • 116
STARTAFT= keyword • 430, 650
  MASSADD function • 430
  REPORT DEVICE function • 650
Statistical Population Counting on Statistics and Diagnostics Area (PXX) Report • 606
Statistics and Diagnostics Area (PXX) • 129, 133, 562, 590, 594
  clearing • 129, 133
  Compound Boolean Selection Facility • 594
  Expression tree dump • 590
  reporting • 562
STATS option • 175, 176, 179
  COMM function • 175, 179
  console command • 176
STATS option (COMM function) • 175
STATUS (DBUTLTY) console command • 37
STATUS console command • 180
STATUS option (COMM function) • 180
STATUS= keyword (ACCT EDIT function) • 76
STIME= keyword (RECOVERY function) • 451
Sub-identifier ID (SID) • 181
Syntax • 35, 301
  DBUTLTY functions • 35
  validating with EDIT function • 301

Syntax diagrams • 26
  reading, how to • 26

T

TABLE= keyword • 254, 323, 341, 430, 437, 450, 494, 550
  CXXMAINT DELETE function • 254
  EXTRACT function • 323
  FILEOUT function • 341
  MASSADD function • 430
  OLREORG function • 437
  RECOVERY function • 450
  REFGROUP report • 550
  REPLACE function • 494
TABLE= keyword (ACCT function) • 76, 86
  EDIT option • 76
  SPILL option • 86
Tables • 225, 228, 253, 321, 399, 427
  adding records • 427
  adding to a data area • 399
  altering logging specification • 225, 228
  deleting definitions • 253
  extracting data from • 321
  loading several into a data area • 399
task information • 160
TASKS= keyword • 376
  INIT FXX function • 376
TERMID= keyword (COMM ALTER function) • 116
TRACE= keyword (COMM ALTER function) • 116, 123
TSN • 688
  resetting • 688
TSN= keyword (INIT LXX function) • 371
TYPE= keyword • 550
  REFGROUP report • 550
TYPE= keyword (REPORT function) • 502, 530, 659
  CXX • 502
  IXX • 530
  Utility Function History • 659
TYPE=B CXX Report • 527
TYPE=I (Impact Report) • 247, 527

U

Unique Row Identifier (URI) • 397
  converting to • 397
UNLOCK function • 719
Update CXX • 675
UPDATE= keyword • 323, 530
  EXTRACT function • 323
  REPORT function - IXX (Index Area) • 530
Utility Function History Report (Type H) • 659

V

Variable log block size • 370
Variables in syntax diagrams • 26
VERIFY= keyword • 364
Verifying index and data areas • 723
VERINDEX function • 723