CA Datacom®/DB

DBUTLTY Reference Guide for z/OS

Version 14.02

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 products:

- CA Datacom®/DB
- CA Datacom® CICS Services
- CA Datacom® Datadictionary™
- 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 Disk™ Backup and Restore (CA Disk)
- CA Dynam®/D Disk Management for z/VSE (CA Dynam/D for z/VSE)
- CA Ideal™ for CA Datacom® (CA Ideal)
- CA IPC
- CA Librarian®
- CA Roscoe® Interactive Environment (CA Roscoe IE)
- CA Scheduler® Job Management (CA Scheduler JM)
- CA SYSVIEW® Performance Management (CA SYSVIEW)
- CA Common Services for z/OS
- CA Workload Automation Restart and Tracking for z/OS Schedulers (CA WA Restart and Tracking); 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 27

CA IPC and CA Ideal Utilities 27
JCL Example Notation 28
Sample Report Headers 28
Reading Syntax Diagrams 30
- Statement Without Parameters 30
- Statement with Required Parameters 31
- Delimiters Around Parameters 31
- Choice of Required Parameters 31
- Default Value for a Required Parameter 32
- Optional Parameter 32
- Choice of Optional Parameters 32
- Repeatable Variable Parameter 33
- Separator with Repeatable Variable and Delimiter 33
- Optional Repeatable Parameters 34
- Default Value for a Parameter 34
- Variables Representing Several Parameters 34

Chapter 2: Using CA Datacom/DB Utility (DBUTLTY) 37

Overview 37
- Multiple Commands 37
- Prerequisite Commands 37
- Syntax Check 38
- Messages 38
- JCL Requirements for z/OS 38
- Requirement to Run APF-Authorized, z/OS 39

Command Syntax 39
- Command 39
- Keyword Parameters 39
- Line Length 39
- End of Command 40
- Comments 40

Console Commands 41
- DBUTLTY STATUS 41

DBUTLTY Dynamic Extend Considerations 43
Impact of Table Partitioning on DBUTLTY Functions 43

Any Parent Table Name Restriction 43
Area level DBUTLTY control 44
BACKUP Utility 44
EXTRACT Utility 44
LOAD Utility 44
REORG Utility 46
Forward Recovery Utility 46
Area Level DBUTLTY Control 46
MULTUSE= Keyword 46
Database ACCESS 49
Area Level ACCESS 51
COMM CLOSE Function 52
Dynamic System Table 52
Summary of Restrictions and Requirements 53
Utility Function Summary 53
Simplify Considerations 57
Functions Requiring Inactive MUF 57
Functions Needing the CXX 62
JCL Requirements for the CXX Data Set 70

Chapter 3: ACCESS (Restricting Opens or Maintenance) 71

Length of Time to Process 71
Buffer Writes 72
Area Level DBUTLTY Control 72
How to Use the Console Command 73
How to Use the DBUTLTY Command 74
Example JCL 79
Sample Report 79

Chapter 4: ACCT (Accounting Facility) 81

Activating the Accounting Facility 81
ACCT CATALOG (Catalog Accounting Database) 82
When to Use 82
How to Use 83
Example JCL 84
Sample Report 84
ACCT CLOSE (Close Accounting Tables) 87
When to Use 87
How to Use the Console Command 88
How to Use the DBUTLTY Command 88
Example JCL 89

Sample Report 89
ACCT EDIT (Edit Accounting Database Definitions) 90
When to Use 90
How to Use 91
Example JCL 93
Sample Report 93
ACCT OPEN (Open Accounting Tables) 95
When to Use 96
How to Use the Console Command 96
How to Use the DBUTLTY Command 97
Example JCL 98
Sample Report 98
ACCT SPILL (Force Accounting Data Recording) 99
When to Use 99
How to Use the Console Command 99
How to Use the DBUTLTY Command 100
Example JCL 101
Sample Report 101

Chapter 5: AUTOCOLL (Collect and Manage Snapshots) 103

Create a Snapshot 103
How to Use 103
Example JCL 105
Sample Report 105
Process a Snapshot Rowset into a Delta Rowset 106
When to Use 106
How to Use 107
Example JCL 108
Sample Report 108
User Created SUMMARY Delta Rowset 109
When to Use 110
How to Use 110
Example JCL 113
Sample Report 113
User Created BASELINE Delta Rowset 114
When to Use 115
How to Use 115
Example JCL 118
Sample Report 118
User Created AVGPERF Delta Rowset 119
When to Use 120

How to Use 120
Example JCL 123
Sample Report 123
Create DSV Output Datasets 124
When to Use 125
How to Use 125
Example JCL 129
Sample Report 130
Sample Report 130
Reporting Snapshot Rowsets 131
When to Use 131
How to Use 131
Example JCL 133
Sample Report 133
Reporting Delta Rowsets 134
When to Use 135
How to Use 135
Example JCL 138
Sample Report 138
Deleting Snapshot Rowsets 139
When to Use 139
How to Use 140
Example JCL 142
Sample Report 142
Deleting Delta Rowsets 144
When to Use 144
How to Use 144
Example JCL 147
Sample Report 147

Chapter 6: AUTOINFO (Collect Environmental MUF Information) 151

Overview 151
Using AUTOINFO in DBUTLTY 151
When to Use 152
How to Use 152
Example JCL 154
Sample Reports 154

Chapter 7: AUTOSTAT (Report on MUF Information) 163

When to Use AUTOSTAT 163
How to Use AUTOSTAT 164

Console Commands....166
MODIFY Command Format....167
Example JCL....169
Sample Report....170

Chapter 8: BACKUP (Create Backups) 171

BACKUP CXX (Back Up Directory)....171
When to Use....171
How to Use....172
Example JCL....174
Sample Report....174
Back Up Data Area....176
When to Use....176
How to Use....181
Example JCL - Native Key Sequence....190
Sample Report....190
Example JCL - Physical Sequence....192
Sample Report....192

Chapter 9: COMM (Communicate with MUF) 195

Overview....195
Length of Time to Process a Request....195
COMM ALTER (Modify Error Table)....196
When to Use....196
How to Use....196
Example JCL....203
Sample Report....203
COMM CLOSE (Close Database or Area)....204
When to Use....204
How to Use the Console Command....204
How to Use the DBUTLTY Command....205
Example JCL....208
Sample Report....208
COMM CLRnnn (Reset Alternate Output Data Sets)....209
When to Use....209
How to Use the Console Command....210
How to Use the DBUTLTY Command....210
Example JCL....212
Sample Report....212
COMM CLRPXX (Clear Statistics and Diagnostics Area (PXX))....213
When to Use....213

How to Use the Console Command 213
How to Use the DBUTLTY Command 214
Example JCL 215
Sample Report 215
COMM CONSOLE (Simulate Console Command) 216
When to Use 216
How to Use the DBUTLTY Command 216
Example JCL 218
COMM EOJ (End MUF Processing) 219
When to Use 219
How to Use the Console Command 221
How to Use the DBUTLTY Command 221
Example JCL 223
Sample Report 223
MUF Startup Options Echoed 224
Summary Reports Produced by the MUF During its EOJ Process 224
COMM NEWRXX (Request New Recovery File) 259
When to Use 260
How to Use the Console Command 261
How to Use the DBUTLTY Command 262
Example JCL 263
Sample Report 263
COMM REQABORT (Abort a Request) 264
When to Use 265
How to Use the Console Command 265
How to Use the DBUTLTY Command 266
Example JCL 268
Sample Report 268
COMM SNAP (Write Dump of Master List) 269
When to Use 269
How to Use the Console Command 269
How to Use the DBUTLTY Command 270
Example JCL 272
Sample Report 272
Master List Dump Example 273
COMM SNAPCSA (Print Dump of Common Storage Area) 273
When to Use 274
How to Use 274
Example JCL 275
Sample Report 275
COMM SNAPSVC (Print Dump of the SVC) 276
When to Use 276

How to Use 276
Example JCL 278
Sample Report 278
Sample Report: COMM SNAPSVC 278
COMM STATS (Write Current Statistics) 278
When to Use 279
How to Use the Console Command 279
How to Use the DBUTLTY Command 280
Example JCL 282
Sample Report 282
COMM STATUS (Print Status of MUF Jobs) 283
When to Use 283
How to Use the Console Command 284
How to Use the DBUTLTY Command 284
Example JCL 286
Sample Report 286
Sample COMM STATUS Online 303
COMM TASK_ABORT (Abort a Task) 304
When to Use 304
How to Use 304
How to Use the DBUTLTY Command 305

Chapter 10: CONFIRM (Verify Constraint Integrity) 307

When to Use 307
Determining Constraints In Effect 308
How to Use 308
Example JCL 310
Sample Report 310

Chapter 11: CXXCLONE (Clone Environment) 313

When to Use 313
How to Use 315
Examples 319
Example JCL 320
Sample Report 320

Chapter 12: CXXMAINT (Maintain Directory) 325

CXXMAINT ALTER CBSUSE (Trial Delete a Key from CBS Use) 326
When to Use 326
How to Use 326

Example JCL 328
Sample Report 328
CXXMAINT ALTER CXXENQ (Controlling ENQ) 329
When to Use 329
How to Use 329
CXXMAINT ALTER DBCS (Activating DBCS Support) 331
When to Use 331
How to Use 331
Example JCL 332
Sample Report 333
CXXMAINT ALTER DSN (Modify Data Set Name) 333
When to Use 333
How to Use 334
Example JCL 336
Sample Report 336
CXXMAINT ALTER DSOP (Modify Space Reclamation Option) 337
When to Use 337
How to Use 337
Example JCL 339
Sample Report 339
CXXMAINT ALTER LANGUAGE (Activating LANGUAGE Support) 340
When to Use 340
How to Use 340
Example JCL 341
Sample Report 341
CXXMAINT ALTER LINK (Disable CXX Name Edit) 342
When to Use 342
How to Use 343
Example JCL 344
Sample Report 345
CXXMAINT ALTER LOGGING (Start/Stop Logging by Table) 345
When to Use 345
How to Use 346
Example JCL 347
Sample Report 348
CXXMAINT ALTER OPTION2= (Activating Other Features) 348
CXXMAINT ALTER OPTION2=datafs (DATAFS Support) 349
CXXMAINT ALTER OPTION2=datahu (High Used Mark) 350
CXXMAINT ALTER OPTION2=areaev (Data Area Extent Validation) 353
CXXMAINT ALTER OPTION2=sumode (Single User Mode) 354
CXXMAINT ALTER OPTION2=MATCHSID 355
CXXMAINT ALTER SINGLE (Run in a Single User Environment) 357

When to Use 357
How to Use 357
Example JCL 358
Sample Report 358
CXXMAINT CONVERT1214 (Upgrade CXX to Current Version) 359
When to Use 359
How to Use 359
Example JCL 361
Sample Report 361
CXXMAINT CONVERT1412 (Downgrade CXX to Prior Version) 362
When to Use 362
How to Use 363
Example JCL 364
Sample Report 364
CXXMAINT DDPROD (Update Directory) 365
When to Use 366
How to Use 369
Example JCL 371
Sample Report 371
CXXMAINT DELETE (Delete CXX Definitions) 373
Deleting a Database 373
Deleting Tables 373
How to Use 374
Example JCL 375
Sample Report 375
Example JCL 376
Sample Report 376
CXXMAINT FORMAT0102 (Base Level Full Features) 377
When to Use 377
How to Use 377
Example JCL 378
Sample Report 378
CXXMAINT FORMAT0201 (Base Level Restrict to r11 Compatible Features) 379
When to Use 379
Considerations for Using FORMAT0201 380
How to Use 380
Example JCL 381
Sample Report 381
CXXMAINT PURGE (Remove Database from Directory) 382
When to Use 382
How to Use 383
Example JCL 383

Sample Report 383

Chapter 13: DBTEST (Execute Test/Analysis Programs) 385

DBTEST Command 1 (DBURSTR) 386
How to Use 386
DBTEST Command 2 (DBURTBL) 388
DBTEST Command 3 (DBUREND) 391
DBTEST Command 4 (DBTEST) 392
DBTEST DBTSTPR (Test Program) 393
When to Use 393
Control Statements 394
Example DBTSTPR JCL 396
Sample Report 396
DBTEST DBIDCPR (Index Analysis Program) 398
When to Use 398
How to Use 399
Example DBIDCPR JCL 399
Sample Report 399
DBTEST DBIDPPR (Index Population Analysis Program) 400
When to Use 400
How to Use 401
Example DBIDPPR JCL 401
Sample Report 402

Chapter 14: DEFAULT (Control Defaults) 405

When to Use 405
How to Use 405
Sample Report 406

Chapter 15: DEFRAG (Defragment Index) 409

Overview 409
Locking 410
Processing 410
Restrictions 411
Stopping DEFRAG 411
Security Considerations 411
DEFRAG 411
When to Use 411
How to Use 411
Example JCL 414

Sample Report 414

Chapter 16: EDIT (Validate Syntax) 417

When to Use 417
How to Use 417
Example JCL 418
Sample Report 418

Chapter 17: ENCRYPT (Facilitate Data Encryption) 421

When to Use 421
Opening and Backing Up Encrypted Tables 422
DBUTLTY LOAD and REORG BASIC Encryption Functions 422
How to Use 423
SET_BASIC_KEY_n Option 423
Example JCL 424
Sample Report 424

Chapter 18: EXTBKUP (Extract from Backup File) 427

When to Use 428
How to Use 428
Example JCL 430
Sample Report 431

Chapter 19: EXTEND (Extend Data or Index Areas) 433

When to Use 433
How Space Is Allocated for an EXTEND 434
Verifying the Extend 435
Successful Execution Requirements and Controls 435
How to Use 436
Example JCL 438
Sample Report 438

Chapter 20: EXTRACT (Extract Data Table) 441

When to Use 441
Extracting in Native Key Sequence 442
Extracting in Physical Key Sequence 442
EXTRACT Output 443
Master List Requirements 443
Successful Execution Requirements and Controls 444

How to Use 445
Parallel Extract 451
FIRSTKEY= and LASTKEY= Keywords in Parallel Extract 452
Restrictions 453
Example JCL 453
Sample Report 454

Chapter 21: FILEOUT (Define Output Data Sets) 455

When to Use 455
How to Use 455
Examples 457

Chapter 22: FLEXPOOL (Dynamic Buffers) 459

When to Use 459
How to Use the Console Command 460
How to Use the DBUTLTY Command 461
Example JCL 462
Sample Report 462

Chapter 23: FORCE (Force Keyword Values) 465

When to Use 465
How to Use 465
Sample Report 466

Chapter 24: GAPFIX (Fix CDC Gaps) 467

When to Use 467
How to Use 469
GAPFIX Reports 471
Sample Report 471
GAPFIX Detail Report Sample 473

Chapter 25: INIT CXX/FXX/IXX/LXX/WXX/Data Area 475

Restrictions 475
Multivolume INITs 475
INIT CXX (Format Directory) 475
When to Use 475
How to Use 477
Example JCL 478
Sample Report 478

INIT Data Area (Format Data Area) 479
When to Use 480
How to Use 482
Example JCL 485
Sample Report 485
INIT IXX (Format Index Area) 486
When to Use 487
How to Use 490
Example JCL 493
Sample Report 494
INIT LXX (Format Log Area) 494
When to Use 495
How to Use 497
Example JCL for Single Log Area 498
Sample Report 498
INIT FXX (Format Force Area) 499
When to Use 500
How to Use 501
Example JCL for Force Area 502
Sample Report 502
INIT WXX (Format RRM Log) 503
When to Use 503
How to Use 504

Chapter 26: LINK (Link to the CXX) 505

When to Use 506
How to Use 507
Example JCL 507
Sample Report 508

Chapter 27: LOAD (Build CXX or Data Area) 511

Overview 511
LOAD CXX (Build CXX) 512
When to Use 512
How to Use 515
Example JCL 517
Sample Report 518
LOAD (Build a Data Area) 519
When to Use 519
Loading Multiple Tables Into a Data Area 523
Adding a Table to a Data Area Not Using REPLACE 524

How to Use 527
Example JCL 544
Sample Report 544
SORTDFLT=YES Report Section for LOAD 551

Chapter 28: LOCK (Lock a Data Area) 555

When to Use 555
How to Use 555
Example JCL 556
Sample Report 557

Chapter 29: MASSADD (Add Records to Table) 559

When to Use 559
Use of Pipeline 559
Log Check Points and Backout 559
Unique Keys 560
Integrity Constraints 560
Use of the MUF 560
Tables Using DBVVRPR 560
Successful Execution Requirements and Controls 561
How to Use 562
Example JCL 565
Sample Report 565

Chapter 30: OLREORG (Reorganize a Database Online) 567

Restrictions 568
Security Considerations 568
When to Use 568
Using Empty Blocks in OLREORG 569
How to Use 570
Example JCL 573
Sample Report 573

Chapter 31: PREINIT (Pre-Format Index and Data Areas) 575

When to Use PREINIT index area 577
Successful Execution Requirements and Controls 578
Multivolume INITs 578
Controlling the Index Block Size 579
How to Use 579

Examples 583
Example JCL 584
Sample Report 584
When to Use PREINIT data area 585
Multivolume INITs 585
Successful Execution Requirements and Controls 586
How to Use PREINIT data area 586
Examples 588
Example JCL 589
Sample Report 589

Chapter 32: RECOVERY (Rebuild a Database) 591

Types of Recovery 591
When to Use 591
How to Use Backward Recovery 596
How to Use Forward Recovery 597
RECOVERY Command 1 (RECBASE) 599
RECOVERY Command 2 (RECJOB) 600
RECOVERY Command 3 (RECOVERY) 602
Backward Recovery 602
Forward Recovery 603
Recovery Examples 613
Example JCL 1 614
Example JCL 2 614
Sample Report 614

Chapter 33: REMOVE (Remove Key ID) 619

When to Use 619
Changing a Key Definition Length 619
How to Use 620
Example JCL 621
Sample Report 621

Chapter 34: REORG (Parallel BACKUP/LOAD) 623

When to Use 623
Parallel Backup in REORG 624
DIAGOPTION 9,16,ON/OFF 626
Parallel Backup Restrictions 626
Parallel Data Reorganization 626
Restrictions for Parallel Data Load and Parallel Index Update 629

Parallel Index Update 630
Post Parallel Reorganize Requirements 631
BACKUPONLY or BACKUP phase of BACKUPLOAD Execution Requirements/Controls 631
LOADONLY or LOAD phase of BACKUPLOAD Execution Requirements/Controls 632
How to Use 633
Example JCL 639
Sample Report 639

Chapter 35: REPLACE (Replace All Data in a Table) 641

Restrictions 641
When to Use 641
Highlights of the Utility Side of the Processing 642
Highlights of the MUF Processing 643
Successful Execution Requirements and Controls 643
How to Use 643
Example JCL 649
Sample Report 649

Chapter 36: REPORT (Generate Reports) 651

Data Area Space Estimation Report (Device) 651
When to Use 651
How to Use 651
Example JCL (3380) 654
Sample Report 654
Directory (CXX) Reports 657
When to Use 658
How to Use 658
Full Directory Report 660
Example JCL: Full CXX Report 660
Sample Report 661
Sample Report: REPORT CXX - Full - Special Fields 676
Type A - Directory Space Utilization 678
Encryption Report without TYPE=A 679
Example JCL: Space Utilization CXX Report 679
Sample Report 679
Type B - CXX Report 681
Type I - Impact Report 681
Impact CXX Report 682
Sample Report 682
Type K - Encryption Report 683
Type P - PREINIT Report 683

Example JCL 684
Sample Report 684
Index Area (IXX) Reports 685
When to Use 686
How to Use 686
Type A - Sequenced by Table Name 688
Example JCL 689
Sample Report 689
Sample Report: REPORT IXX - Type A 689
Type B - Sequenced by Key ID 690
Example JCL 691
Sample Report: REPORT IXX - Type B 691
Types D and E - Dump of the Index Area 692
Example JCL 693
Sample Report: REPORT IXX - Type D 693
Type G - Number of Records With Unique Key Values 695
Example JCL 695
Sample Report: REPORT IXX - Type G 695
Log Area (LXX) Report 697
When to Use 698
How to Use 698
Example JCL 699
Sample Report 699
Recovery File (RXX) Report 701
Recovery File (RXX) Analysis Report 702
Example JCL 705
Sample Report 705
Split File Analysis Report 714
How to Use 714
REFGROUP Report 720
Restrictions 720
When to Use 720
How to Use 721
Example JCL (Summary REFGROUP Report TYPE=S) 723
Sample Report 723
Sample Report: REFGROUP Summary 723
Example JCL (Detail REFGROUP Report TYPE=D) 725
Sample Report 725
Sample Report: REFGROUP Detail 726
Example JCL (Detail REFGROUP Report TYPE=K) 727
Sample Report 728
Sample Report: REFGROUP Detail with Key Value Information 728

REPORT MEMORY=CF (Coupling Facility) 730
REPORT MEMORY=DS (Data Space) 730
When to Use 730
How to Use 731
Example JCL 731
Sample Report 731
REPORT MEMORY=MVS 732
When to Use 732
How to Use 732
Example JCL 733
Sample Report 734
Statistics and Diagnostics Area (PXX) Report 739
When to Use 739
How to Use 741
Generating a PXX Report 745
Example JCL 745
Sample Report 746
Expression Dump 761
Sample Report 765
Reporting on the Compound Boolean Selection Facility 765
Example JCL 766
Sample Report 766
Sample Report: REPORT PXX - CBS Facility Diagnostic Report – Example 1 766
Sample Report: REPORT PXX - CBS Facility Diagnostic Report – Example 2 771
Statistical Population Counting Report 775
Dumping SQL Internal Control Structures 777
TYPE=DATANE - Native Sequence Efficiency Report 778
Restrictions 779
When to Use 779
How to Use 779
Example JCL: DATANE without RECORDS= Specified 781
Sample Report: DATANE without RECORDS= Specified 782
Example JCL: DATANE with RECORDS= Specified 785
Sample Report: DATANE with RECORDS= Specified 785
Example JCL: DATANE for a Compressed Table with NUMBER=0200 Specified 788
Sample Report: DATANE for a Compressed Table with NUMBER=0200 Specified 789
TYPE=DATASP - Data Area Space Report 791
Restrictions 792
When to Use 792
How to Use 792
Example JCL: DATASP without BLOCKS= Specified 794
Sample Report: DATASP without BLOCKS= Specified 794

Example JCL: DATASP with BLOCKS= Specified 796
Sample Report: DATASP with BLOCKS= Specified 797
TYPE=H - Utility Function History Report 799
When to Use 799
How to Use 799
Example JCL 800
Sample Report 800
TYPE=U - Debugging Report 804
When to Use 805
Security Considerations 805
How to Use 805
Examples 807

Chapter 37: RESET (Reset Area) 811

RESET CXX (Release Directory and Database Control) 811
When to Use 811
How to Use 812
Example JCL 814
Sample Report 814
RESET LXX (Reset Log Area to Empty) 815
When to Use 815
How to Use 815
Example JCL 816
Sample Report 816

Chapter 38: RETIX (Reconstruct Index) 819

Rebuilding the Data Space Index or Recounting Records 819
When to Use 820
How to Use 821
Rebuilding an Index Area 822
When to Use 823
How to Use 825
Example JCL 833
Sample Report 833
SORTDFLT=YES Report Section for RETIX 835
Resetting the TSN 837
When to Use 838
How to Use 840

Chapter 39: RXXFIX (Recover End-Of-File) 843
When to Use ... 843
RESTART Processing of a Tape Without End-of-File ... 843
Update SPILL History Table ... 843
How to Use ... 844
Example JCL ... 845
Sample Report ... 846
Chapter 40: SECURITY RESET (Clearing Security Validation Buffer) 847
When to Use ... 847
How to Use the Console Command ... 847
How to Use the DBUTLTY Command ... 848
Example JCL ... 849
Sample Report ... 849
Chapter 41: SET (Change Environment Processing) 851
When to Use ... 853
How to Use ... 853
Example JCL ... 861
Sample Report ... 862
Chapter 42: SPILL (Transfer Data to RXX) 863
When to Use Active Recovery ... 863
When to Use Inactive Recovery ... 864
Dual Recovery File (RXX) Support ... 864
MUF Connection ... 864
How to Use ... 865
Example JCL ... 865
Sample Report ... 866
Chapter 43: SPILLOPT (Transfer Data to RXX Using MAX/MIN) 867
Update SPILL History Table ... 867
How to Use ... 867
Example JCL ... 869
Sample Report ... 869
Chapter 44: UNLOCK (Unlock a Data Area) 871
When to Use ... 871

How to Use 871
Example JCL 872
Sample Report 872

Chapter 45: VERINDEX (Verify Index) 875

When to Use 875
How to Use 875
Data to Index (DATAINDEX) 876
Sample Report 877
Sample Report: DATAINDEX without an Error 877
Sample Report: DATAINDEX with an Error 878
Index to Data (INDEXDATA) 878
Sample Report 879
Example Dump: INDOUBT Index Entries 880

Chapter 1: Introduction

This reference guide describes the various CA Datacom/DB Utility (DBUTLTY) functions and gives instructions for using each of them. This guide is a companion manual to the *CA Datacom/DB Database and System Administration Guide*. CA Datacom Datadictionary utilities are documented in the *CA Datacom Datadictionary Batch Reference Guide*.

This guide refers you to other manuals for additional details about specific tasks. Also, in your support of other organizations, having copies of the manuals they use is often necessary.

This guide is primarily intended for Database Administrators and systems programmers responsible for:

- Administering the operations of the CA Datacom and CA Ideal system
- Maintaining data integrity and accessibility
- Installing and supporting the CA system software
- Managing and implementing definitions
- Performing routine maintenance tasks

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*

Product	Utility Name	Function	Refer to:
CA Ideal	IDUTOTRN	Object Transport Utility	*CA Ideal Working in the Environment Reference Guide*
CA Ideal	IDUTSTRN	Source Transport Utility	*CA Ideal Working in the Environment Reference 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.

Sample Report Headers

The report headers for the sample reports contained in this guide are shown here. Report headers have the following format:

```
Date: mm/dd/ccyy    ********************************************************************************        Page:     n
                    *                                CA Datacom/DB                                 *
Time: hh.mm.ss      *                              General Utility                                 *      Version: nn.n
                    *                Copyright © 1990-2011 CA. All rights reserved.                *
Base:     dbid      ******************************************************************************** Directory:    name
```

Base:

The *dbid* is the DATACOM-ID (DBID) of the database (base) in use when the report was executed.

Note: Base does not appear in the header if it is not appropriate for the report that was generated or not known at the time the report is produced.

Date:

The date when the report was executed is shown in the format *mm/dd/ccyy*:

mm

month

dd

day

cc

century

yy

year

Directory:

The *name* is the internal name of the Directory (CXX), assigned with the INIT CXX function that was in use when the report was executed and if known at the time the report was produced.

Note: Directory does not appear in the header if it is not appropriate for the report that was generated.

Page:

The *n* is the page number of the report.

Time:

The time when the report was assembled is shown in the format *hh.mm.ss*:

hh

hour

mm

minutes

ss

seconds

Version:

The version of CA Datacom/DB being executed when the report was executed is shown in the format *nn.n,* for example, Version: 14.0.

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.

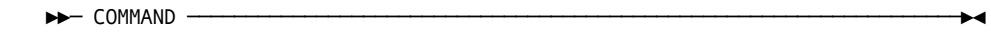
The following terminology, symbols, and concepts are used in syntax diagrams:

- 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 and variables are indicated by 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:



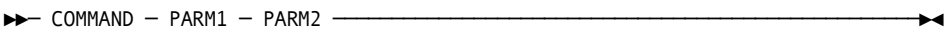
For this statement, you must write the following:

```
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:



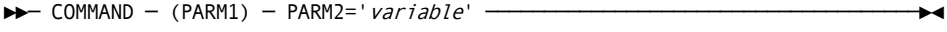
You must write the following:

```
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:



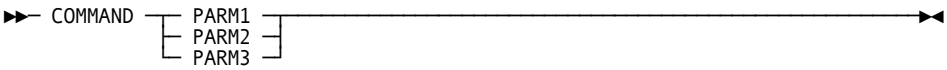
If the word *variable* is a valid entry, you must write the following:

```
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:



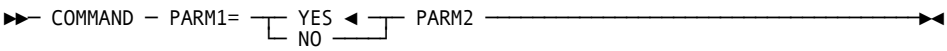
You can choose one of the parameters from the vertical list, such as in the following examples:

```
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 is identified by an arrowhead pointing to the left. The default value is the value automatically assigned to 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:



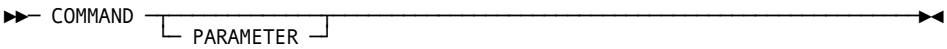
If you specify the command, you must write one of the following:

```
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:



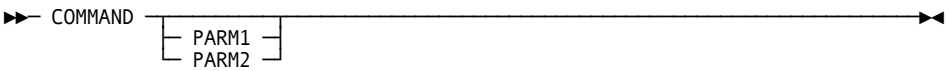
You can choose (or not) to use the optional parameter, as shown in the following examples:

```
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:



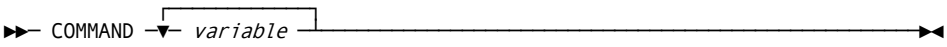
You can choose any of the parameters from the vertical list, or you can write the statement without an optional parameter, such as in the following examples:

```
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:



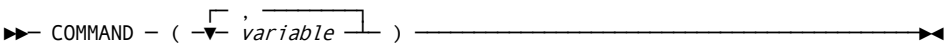
In the preceding diagram, the word *variable* is in lowercase italics, indicating that it is a value you supply, but it is also on the main path, which means that you are required to specify at least one entry. The repeat symbol indicates that you can specify a parameter more than once. Assume that you 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:



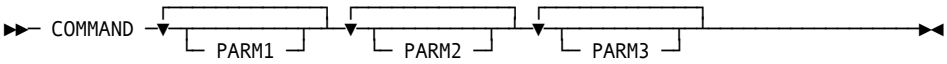
In the preceding diagram, the word *variable* is in lowercase italics, indicating that it is a value you supply. It is also on the main path, which means that you must specify at least one entry. The repeat symbol indicates that you can specify more than one variable and that you must separate the entries 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:



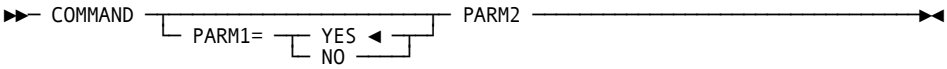
The following are some of the statements you can write:

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

Default Value for a Parameter

The arrowhead pointing to 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:

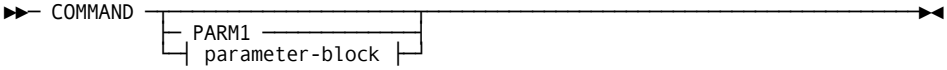


For this command, COMMAND PARM2 is the equivalent of COMMAND PARM1=YES PARM2.

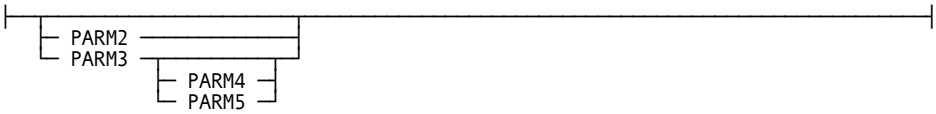
Variables Representing Several Parameters

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, you must specify PARM3.

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

Overview

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 guide 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. 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.

In z/OS environments the Multi-User Facility (MUF) and the CA Datacom/DB Utility (DBUTLTY) must run authorized. The MUF is authorized when all JOBLIB/STEPLIBs are authorized. DBUTLTY can run some of the functions not APF authorized, but the intent of DBUTLTY is to run authorized.

For more information, see the *CA Datacom/DB Database and System Administration Guide*.

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 for z/OS

If snapping is necessary, CA Datacom/DB looks for a DD statement with a name of SNAPER. If it is not provided, CA Datacom/DB dynamically allocates a DD statement for SNAPER with the value SYSOUT=*. Unless you want a specific SYSOUT class or other options, do not provide the DD statement. CA Datacom/DB also dynamically allocates a SYSPRINT DD statement when needed if you do not provide it. The SYSPRINT DCB BLKSIZE and LRECL defaulted or are set based upon the DBSIDPR module assembled with the DBSYSID macro with keyword LINE=NEW or LINE=OLD. With LINE=NEW (the default) the BLKSIZE defaults to 0 if not set and the LRECL defaults to 133 if not set. With LINE=OLD, the BLKSIZE defaults to 121 if not set and the LRECL is set to 121, overriding any value provided. With LINE=OLD, if you specify a different block size in the JCL, it must be a multiple of 121. The exception to this is the COMM OPTION=SNAPCSA. This utility requires the SNAPER DD statement. During certain errors or on request, DBUTLTY can direct an address space snap to DD statement SNAPAS*nn*, where *nn* is a sequential number such as 01.

Requirement to Run APF-Authorized, z/OS

We recommend (and DBUTLTY always expects to be) running authorized.

Certain DBUTLTY functions require that the job run APF-authorized. See the documentation on the individual functions to know if it requires APF-authorization.

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
             DDNAME=INPUT                         From ddname 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
                DDNAME=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 MUF STATUS. The option is controlled by keyword CONSOLE_MINUTES= in the DBSYSID macro that generates the DBSIDPR loaded module.

To support the console status of DBUTLTY, do not specify a 0 (zero) for CONSOLE_MINUTES=. Specifying CONSOLE_MINUTES=0 means that you do not want console support. We do not recommend using CONSOLE_MINUTES=0. Specifying any value other than 0 allows the console command STATUS to be issued to an executing DBUTLTY as a z/OS 'modify' command, and DBUTLTY responds.

If you do not specify CONSOLE_MINUTES=0, DBUTLTY attaches a subtask during its startup and detaches that subtask during its ending process. The subtask supports a console STATUS command and other allowed console commands. DBUTLTY functions that include status support are: BACKUP, EXTRACT, LOAD, REORG, and RETIX.

The default is CONSOLE_MINUTES=10, indicating that every 10 minutes of execution, DBUTLTY provides an automatic status. With CONSOLE_MINUTES= not equal to 0, 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.

Note: The option to select an interval of time is only available in z/OS environments. Also, the DBFLSUB callable load process is not part of DBUTLTY and does not support the console status option.

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

More information about these messages can be found in 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, the LOAD function attempts to dynamically extend the existing extents, with restrictions. Details of the dynamic extend match those that would occur in the 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 CA Datacom 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 dynamic extend feature is 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 CA Datacom 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 more information about the DYNAMIC_EXTEND console-like command, see the *CA Datacom/DB Database and System Administration Guide*.

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).

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

The following utility functions have special changes because of partitioning.

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.

EXTRACT Utility

The extract utility function allows the Full Parent name. Three asterisks (***) should be coded in AREA= if you are extracting a partition Full Parent table. One specific table name is supported.

LOAD Utility

For a LOAD with FORMAT=BACKUP, you must use the 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-partitioned 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.

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 DDNAME=*,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 MUF environment.

This feature requires that the database be optimized in the MUF, which is done by *not* having the ACCESS MUF 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 MUF and Single User modes. The database involved must be able to be opened for update in the MUF 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 MUF. The actual data set open is performed in the utility region, and the JCL requirements are applied to the utility region.

  Note: The operating system performs the catalog portion of the DD statement DISP option when the DD statement is removed. This occurs at job step termination. Therefore, if you were to create a new data set with an INIT, or to change the volume list during an extend, do not expect the MUF to correctly open this area until after the step completes. This restricts the DBUTLTY functions that succeed when following a catalog change.

  When you specify or allow the default of MULTUSE=NO in the INIT or EXTEND functions for data areas, the functions execute without area level control in MUF, 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 MUF and Single User modes. The database involved must be able to be opened for update in the MUF 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 MUF to close it, so that any type of open data set lock can be released, which could be necessary 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 MUF.

  With MULTUSE=YES:

  - The area is first opened in MUF 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 MUF address space. Packages of index entries (about 32k bytes) are sent from the utility to the MUF.

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. The following options must be in effect: URI=YES, OPTIMIZE=NO, VERIFY=NO, and LOADDUPS=NO.

When MULTUSE=NO is specified or is allowed to default in the LOAD and RETIX functions, the functions execute without area level control in MUF, that is, the database involved may not be open for update, and all processing executes in the Single User mode.

- RECOVERY (forward only)

  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. Other forward recovery functionality executes in Single User mode.

  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 MUF. 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.

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

  - Database must be open for update in the MUF 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 MUF. 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 MUF. When MULTUSE=YES, the index phase must occur in MUF 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 MUF, with the exception of the RESTART process done during the MUF 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 three processes:

- MUF 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 Datadictionary.

The database level of ACCESS is typically done to stop, in an orderly fashion, access in a MUF 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 without Simplify do not know about the ACCESS status of any MUF, nor are they dependent upon any current status.

Following is an example of database ACCESS used to reorganize an entire database, without Simplify. In the example, the DBUTLTY functions BACKUP, INIT, and LOAD each execute in Single User mode with no knowledge of any MUF or its ACCESS status, and therefore do not require them. However, ACCESS helps ensure correct execution of the utility set because it verifies 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 MUF, with the exception of the RESTART process done during the MUF 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 MUF area. It usually precedes a DBUTLTY function at the data area level that needs to execute in the MUF.

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 MUF with no loss of access to other areas in the MUF. In the example, the utility functions BACKUP and LOAD each execute in the MUF. 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 MUF, 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 that the DBID= keyword be specified. You would need to use this feature only if the MUF has the area open and you want to initialize, extend, or load it through DBUTLTY and uncatalog it.

The ability to CLOSE an area does not exist as a console command.

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

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 with the MUF active. 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 MUF. Only the STATUS, SNAPCSA, and SNAPSVC options can be executed in both Single User mode and with an active MUF.
CONFIRM		Verifies that the data in a specific table adheres to SQL-defined integrity constraints.
CXXMAINT		Provides functions to maintain the Directory (CXX).
CXXCLONE		Clones an environment to a new environment that is modeled after an existing environment.

Function	SU	Description
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.
ENCRYPT		Facilitates the data encryption feature of CA Datacom/DB by using certain designated options.
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.
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).

Function	SU	Description
LOCK		Locks a data area when you request a backup while the MUF 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 an active MUF 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 with the MUF active.
REMOVE		Deletes a key ID from the Index.
REORG		Provides a way to reorganize a large data area into *nearly 100 percent* Native Key sequence. The REORG function executes much faster than other reorganization options.
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 MUF and allows full read and update access to every other table in the same area and database to other applications in the same MUF.

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 ■ SVC report ■ Utility Function History Report (TYPE=H)
	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 MUF 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.

Function	SU	Description
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.

Simplify Considerations

Important! For details about the Simplify feature, see the *CA Datacom/DB Database and System Administration Guide*.

Functions Requiring Inactive MUF

The execution of the first function that requires the CXX be available ensures that DBUTLTY has not already connected to a MUF and, if so, it is an error, is reflected by the following message, and undergoes error termination.

DB10053E – MUF RUNNING, FUNCTION REQUIRES MUF DOWN

If not connected, the function removes any DELAY68 setting and attempts to connect to MUF. If it is successful, it is an error, receives a DB10053E message, and undergoes error termination.

Once it is determined that the MUF is not enabled, the DBUTLTY ensures that the MUF cannot be enabled using this CXX. By using a shared global ENQ, it protects the execution of the entire DBUTLTY function set. Execution of a MUF during one of these DBUTLTY executions generates a startup error DB00205E with 1095 CXX DSN OWNED ELSEWHERE. If the ENQ is not available, the error is recognized as the following error message and the function terminated.

DB10055E - A MUF RUNNING WITH CXX, NOT ALLOWED

All functions other than INIT AREA=CXX checks that they are executing on a system in the same Sysplex as the one on which the CXX was initialized and, if not, generates error message DB00912E and return code 47(008). This protection ensures that the GRS ENQ processing protects the data sets and actions being performed.

Specific DBUTLTY functions and their descriptions subject to the rule that MUF is required to not be enabled are as follows:

- INIT AREA=CXX
- LOAD AREA=CXX(no DBID keyword)
- CXXCLONE (no DBID keyword)
- CXXMAINT OPTION=CONVERT1214
- CXXMAINT OPTION=CONVERT1412
- CXXMAINT OPTION=MATCHSID
- INIT AREA=LXX
- INIT AREA=FXX
- RESET AREA=LXX
- RESET AREA=CXX
- SET OPTION1=MUF_NOT_ENABLED (all functions that follow)

INIT AREA=CXX

The INIT AREA=CXX function sets the CXX being built to be in the Simplify mode. If needed, the INIT AREA=CXX function must be the first of these functions that execute in the DBUTLTY step. The INIT AREA=CXX without Simplify recognizes an error if DATACOM=AD was specified and is reflected in the following message and error termination:

DB10057E - REQUIRES DBSIDPR SIMPLIFY YES

The INIT AREA=CXX function checks that the CXX has not been opened by a previous function. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the CXX and also the DISP=OLD protection for the CXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the DBUTLTY step.

Note: When running without Simplify mode, this function trusts that the user executes the function when and where appropriate. The function provides no protection that the data set being initialized is not currently in use by one or more MUF/DBUTLTY executions. The function provides no protection that the data set being initialized does not become in use during any point of the initialization or following functions by one or more MUF/DBUTLTY executions. If executed in error, there are a significant number of error conditions that are possible.

LOAD AREA=CXX(no DBID= keyword)

The LOAD AREA=CXX(no DBID=keyword) function restores the entire CXX from the backup. It does not restore the CXX name and does not alter the Simplify option from the current CXX. This function is normally used in the same DBUTLTY step as the INIT AREA=CXX immediately following. This function can execute in a step following the INIT AREA=CXX provided that the CXX has not been opened by a previous function in this step.

In all cases, the function only executes if the CXX is still empty from the INIT. If the function is not in the INIT AREA=CXX step, it checks that the CXX has not been opened by another function. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the CXX and also the DISP=OLD protection for the CXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the JCL stream.

CXXCLONE (no DBID=keyword)

The CXXCLONE function performs as a special case of LOAD AREA=CXX and has a similar set of rules.

When the DBID is not provided, this function restores the entire CXX from the backup. It does not restore the CXX name or alter the Simplify option from the current CXX. This function is normally used in the same DBUTLTY step as the INIT AREA=CXX immediately following. This function can execute in a step following the INIT AREA=CXX provided that the CXX has not been opened by a previous function in this step.

In all cases, the function only executes if the CXX is still empty from the INIT. If the function is not in the INIT AREA=CXX step, it checks that the CXX has not been opened by another function. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the CXX and also the DISP=OLD protection for the CXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the JCL stream.

Of special note, the CXXCLONE function accepts input as a backup of a Version 14.0 CXX and also a Version 12.0 CXX. This ability allows the migration of a single base or set of bases from Version 12.0 to Version 14.0 not using the CXXMAINT OPTION=CONVERT1214 ability that requires action for every base on the Version 12.0 backup provided.

CXXMAINT OPTION=CONVERT1214 **(or CONVERT1412)**

The CXXMAINT OPTION=CONVERT1214 function loads the entire CXX from the Version 12.0 backup. However, it does *not* alter the CXX name or the Simplify option from the current CXX. This function must occur in the same DBUTLTY step as the INIT AREA=CXX and the CXX must still be empty from the INIT. This function can execute in a step following the INIT AREA=CXX provided that the CXX has not been opened by a previous function in this step and the CXX is still empty from the INIT function.

If the function is not in the INIT AREA=CXX step, it checks that the CXX has not been opened by another function. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the CXX and also the DISP=OLD protection for the CXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the JCL stream.

CXXMAINT OPTION=ALTER,OPTION2=MATCHSID

- The CXXMAINT OPTION=ALTER,OPTION2=MATCHSID function alters the CXX to match the current DBSIDPR definition as related to the DBSYSID macro parameter SIMPLIFY_MODE=.

It executes under the rules of the DBSIDPR module. The function must be the first function to require access to the CXX because it checks that the CXX has not been opened by another function. A violation of this rule is reflected by the following message and error termination:

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the CXX and also the DISP=OLD protection for the CXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the JCL stream.

INIT AREA=LXX

The INIT AREA=LXX function initializes the LXX control area and checks that the LXX has not been opened by another function. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the LXX and also the DISP=OLD protection for the LXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the DBUTLTY step.

Note: When running without Simplify mode, this function trusts that the user executes the function when and where appropriate. The function provides no protection that the data set being initialized is not currently in use by one or more MUF/DBUTLTY executions. The function provides no protection that the data set being initialized does not become in use during any point of the initialization or following functions by one or more MUF/DBUTLTY executions. If executed in error, there are a significant number of error conditions that are possible.

INIT AREA=FXX

The INIT AREA=FXX function initializes the FXX and checks that the FXX has not been opened by another function. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the FXX and also the DISP=OLD protection for the FXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the JCL stream.

Note: When running without Simplify mode, this function trusts that the user executes the function when and where appropriate. The function provides no protection that the data set being initialized is not currently in use by one or more MUF/DBUTLTY executions. The function provides no protection that the data set being initialized does not become in use during any point of the initialization or following functions by one or more MUF/DBUTLTY executions. If executed in error, there are a significant number of error conditions that are possible.

RESET AREA=LXX **and AREA=CXX**

The RESET AREA=LXX function resets the LXX control area to be empty. It is rarely used and should be more rare with Simplify. If used, it must always be done with the MUF not enabled. The RESET AREA=CXX functions reset the CXX control area to have no DASD locking. The database is also reset to not be open and not load or unload. It is not needed if MUF is enabled. If used, it must always be done with the MUF not enabled.

This function checks that the LXX or CXX has not been opened by another function. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the LXX or CXX and also the DISP=OLD protection for the LXX or CXX is required. Both protections remain for the entire step.

This function expects to run with MUF not enabled. Therefore, the function SET OPTION1=MUF_NOT_ENABLED is not required. This function also effectively activates the SET option for the remainder of the JCL stream.

Note: When running without Simplify mode, this function trusts that the user executes the function when and where appropriate. The function provides no protection that the data set being initialized is not currently in use by one or more MUF/DBUTLTY executions. The function provides no protection that the data set being initialized does not become in use during any point of the initialization or following functions by one or more MUF/DBUTLTY executions. If executed in error, there are a significant number of error conditions that are possible.

SET **OPTION1=MUF_NOT_ENABLED (all functions that follow)**

This SET option may only be executed if the DBUTLTY is not already connected to a MUF and has not already opened the Directory CXX. It has affect for the next function that requires the Directory CXX and ensures that MUF is not enabled and that the CXX is used locally. A violation of this rule is reflected by the following message and error termination.

DB10054E - FUNCTION REQUIRES xxx NOT TO BE OPEN

Exclusive ENQ protection of the CXX and also the DISP=OLD protection for the CXX is required. Both protections remain for the entire step.

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

Functions Needing the CXX

With the Simplify feature, normal DBUTLTY functions can access and update control data and user data using MUF. This ensures MUF has control while providing current and updated information.

This section reviews the DBUTLTY functions that need the CXX and in releases prior to Version 14.0 caused the CXX to be opened for update in the DBUTLTY address space. With the Simplify feature, these functions have been changed to use the MUF for all CXX access and update while providing another level of control that benefits the functions.

Updates include the following:

- The CXX DD statement required by these functions can be removed. If CXX update access is no longer needed, you can remove that right from security. This action makes the CXX much safer by having fewer people with the ability to accidentally overlay or change it.

  Note: The CXX DD statement is dynamically allocated if the MUF is down and a function is running that requires MUF to be down or if you have directed that the DBUTLTY is to execute with MUF down.

- A related change that affects all the functions is the Run Unit. Without Simplify, a DBUTLTY function connected to the MUF obtains a Run Unit from the MUF. If that DBUTLTY opens the CXX locally, it also obtains a Run Unit locally. With Simplify, the MUF Run Unit is shared locally and both have a single number.
- All editing of CXX information allowing the function to continue occurs in the MUF address space. All setting of CXX information such as loading, unloading, loaded, and so forth, occurs in the MUF address space.
- All CXX information used is current.

Following are the function changes:

BACKUP

- AREA=CXX gets all information from the MUF. Before processing each database for information, an internal COMM STATS is done to synchronize the current MUF information.
- AREA=a (user data area) communicates with MUF to ensure it is appropriate to execute the function. It is protected by MUF to guarantee no conflict of the BACKUP requirements. When MULTUSE=YES is specified, where allowed, running with Simplify mode works the same as without. For all other cases, Simplify works differently as described following.

  With SEQ=PHYSICAL,UPDATE=NO:

  - Allows execution with the area being open for read or update by other users and produces a hot backup.
  - Causes MUF to push current pipeline buffers from MUF memory to DASD where this backup gets its data.

With SEQ=NATIVE or SEQ=PHYSICAL,UPDATE=YES:

- Ensures no other users of the database exist and once started, allows no other users of the database.
- Attempts to close the database.
- Native obtains a share ENQ on the index areas. Both native and physical obtain a share ENQ on the data areas. To allow a MUF failure and restart to know about the execution of the DBUTLTY function, the DBUTLTY obtains a share ENQ on CADTCM02 and DBUTLTY/cxxname/dbid. This includes all forms of BACKUP and EXTRACT except SEQ=PHYSICAL,UPDATE=NO. If MUF restarts, it uses an ENQ with TEST to see if the DBUTLTY function is still executing. If not, the status is removed. If yes, the base is set to be in open with backup in process running. The function communicates with MUF briefly during its startup processing and then executes within the DBUTLTY address space to read and write all the appropriate rows and then communicate with MUF briefly to note completion.

  When a large number of rows need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the row processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of row processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 (or a DBUTLTY override with SET OPTION1=DELAY68=n) to wait for the enable a set maximum number of minutes.

  Note: This synchronization process occurs once and only at the point between row processing completes and close processing starts.

CXXMAINT

The CONVERT1214, CONVERT1412, OPTION=ALTER,and OPTION2=MATCHSID, functions cannot run with MUF enabled. The other functions expect to run with the MUF enabled where it is used.

EXTBKUP

Does not use the CXX or MUF.

EXTEND

AREA=IXX

- Ensures that the database is not open for any other purpose and once started, blocks other opens.
- Obtains exclusive ENQ of index areas.
- Obtains an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid.
- Communicates with MUF briefly during its startup processing, executes within the DBUTLTY address space to format new tracks and then communicates with MUF briefly to note completion.

  If a large number of tracks need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the track processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of track processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 or a DBUTLTY override with SET OPTION1=DELAY68=n to wait for the enable a set maximum number of minutes.

  Note: This synchronization process occurs once and only at the point between track processing completes and close processing starts.

AREA=a (user data area)

With MULTUSE=YES works the same with Simplify as without Simplify. Without MULTUSE=YES:

- Ensures that the database is not open for any other purpose and once started, blocks other opens.
- Obtains exclusive ENQ of data areas.
- Obtains an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid.
- Communicates with MUF briefly during its startup processing, executes within the DBUTLTY address space to format tracks and then communicates with MUF briefly to note completion.

If a large number of tracks need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the track processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of track processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 or a DBUTLTY override with SET OPTION1=DELAY68=n to wait for the enable a set maximum number of minutes.

Note: This synchronization process occurs once and only at the point between track processing completes and close processing starts.

EXTRACT

Same as BACKUP for data area.

INIT

AREA=IXX

- Ensures that the database is not open for any other purpose and once started, blocks other opens.
- Obtains exclusive ENQ of index areas.
- Sets the CXX database as in INIT INDEX.
- Obtains an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid/IXX/COMMON.
- Communicates with MUF briefly during its startup processing, executes within the DBUTLTY address space to format tracks and then communicates with MUF briefly to note completion.

  If a large number of tracks need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the track processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of track processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 or a DBUTLTY override with SET OPTION1=DELAY68=n to wait for the enable a set maximum number of minutes.

  Note: This synchronization process occurs once and only at the point between track processing completes and close processing starts.

AREA=a (user data area)

With MULTUSE=YES, works the same with Simplify as without Simplify. Without MULTUSE=YES:

- Ensures that the database is not open for any other purpose and once started, blocks other opens.
- Obtains exclusive ENQ of data areas.
- Obtains an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid.
- Communicates with MUF briefly during its startup processing, executes within the DBUTLTY address space to format some tracks and then communicates with MUF briefly to note completion.

  It is unlikely that an INIT of a data area will execute long because it does not format all tracks but just some at the start of the extent. The following LOAD will format the rest. After the track processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of track processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 or a DBUTLTY override with SET OPTION1=DELAY68=n to wait for the enable a set maximum number of minutes.

  Note: This synchronization process occurs once and only at the point between track processing completes and close processing starts.

LOAD

With MULTUSE=YES, works the same with Simplify as without Simplify. Without MULTUSE=YES:

- Ensures that the database is not open for any other purpose and once started blocks other opens.
- Gets an exclusive ENQ of each index area and data area.
- Gets an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid.
- Communicates with MUF briefly during its startup processing and then will execute within the DBUTLTY address space to read and write all the appropriate rows and then communicate with MUF briefly to note completion.

When a large number of rows need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the row processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of row processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 (or a DBUTLTY override with SET OPTION1=DELAY68=n) to wait for the enable a set maximum number of minutes.

Note: This synchronization process occurs once and only at the point between row processing completes and close processing starts.

RECOVERY forward

With MULTUSE=YES works the same with Simplify as without Simplify. Without MULTUSE=YES:

- Ensures that the database is not open for any other purpose and once started, blocks other opens.
- Obtains an exclusive ENQ of each index area and data area.
- Obtains an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid.
- Communicates with MUF briefly during its startup processing, executes within the DBUTLTY address space to read and write all the appropriate rows and then communicates with MUF briefly to note completion.

  When a large number of rows need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the row processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of row processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 (or a DBUTLTY override with SET OPTION1=DELAY68=n) to wait for the enable a set maximum number of minutes.

  Note: This synchronization process occurs once and only at the point between row processing completes and close processing starts.

REORG load portion

With MULTUSE=YES works the same with Simplify as without Simplify. Without MULTUSE=YES:

- Ensures that the database is not open for any other purpose and once started, blocks other opens.
- Obtains an exclusive ENQ of each index area and data area.
- Obtains an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid.
- Communicates with MUF briefly during its startup processing, executes within the DBUTLTY address space to read and write all the appropriate rows and then communicates with MUF briefly to note completion.

  When a large number of rows need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the row processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of row processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 (or a DBUTLTY override with SET OPTION1=DELAY68=n) to wait for the enable a set maximum number of minutes.

  Note: This synchronization process occurs once and only at the point between row processing completes and close processing starts.

REPORT

- AREA=CXX obtains all information printed from the MUF. Before processing each database for information, an internal COMM STATS is done to synchronize the current MUF information.
- AREA=LXX obtains all information printed from the MUF. Does not open the LXX locally.

RESET

Not allowed either LXX or CXX

RETIX

With MULTUSE=YES works the same with Simplify as without Simplify.

Without MULTUSE=YES, it ensures that the database is not open for any other purpose and once started, blocks other opens.

- Obtains an exclusive ENQ of each index area and data area.
- Obtains an exclusive ENQ on CADTCM02 and DBUTLTY/cxxname/dbid.
- Communicates with MUF briefly during its startup processing, executes within the DBUTLTY address space to read and write all the appropriate rows and then communicates with MUF briefly to note completion.

When a large number of rows need processing, the time the DBUTLTY is not talking to MUF can be lengthy. After the row processing and before the close processing, the DBUTLTY has special code to verify it is in synchronization with the MUF and if not, drives a resynchronization process. If there is a MUF outage during the time of row processing and the MUF reenabled prior to the close time, the DBUTLTY automatically disconnects from the failed MUF and reconnects to the current MUF. If the MUF is not currently enabled, the DBUTLTY function honors a DBSIDPR setting of DELAY68 (or a DBUTLTY override with SET OPTION1=DELAY68=n) to wait for the enable a set maximum number of minutes.

Note: This synchronization process occurs once and only at the point between row processing completes and close processing starts.

JCL Requirements for the CXX Data Set

The examples in this guide provide the CXX data set for functions that always, or sometimes, require it. However, to simplify JCL and to provide additional protection of the CXX, a best practice is to always omit the JCL for the CXX if it is not creating the data set (DISP=NEW). Instead, use the DBSIDPR module assembled with the DBSYSID macro keyword DSN_XXX=, specifying a model data set name for the CXX, LXX, and FXX that allows DBUTLTY (and MUF) to dynamically allocate the CXX only when it is needed.

One of the large benefits of Simplify mode is that the CXX is opened and processed in MUF with a few exceptions. With the commitment that MUF will always be used, update security to the CXX can be removed from all but a few DBA type users at a site. This will increase the safety of the CXX significantly.

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 MUF is active. You can also use it to allow or disallow the ability of applications to execute maintenance to a database while MUF is active. In Simplify mode, all DBUTLTY functions use MUF when enabled for CXX access. Function options that, without Simplify, use the CXX in MUF and honor ACCESS OFF/READ continue with Simplify to honor ACCESS OFF/READ. However, in Simplify mode for functions that without Simplify use the CXX in the user address space, and now instead use the MUF address space, must completely ignore the ACCESS setting. Each DBUTLTY ensures that it executes only when appropriate and prevents users or other DBUTLTY functions from execution in conflict.

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.

The Access function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, help ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information about XCF.

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 MUF 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

The 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 MUF applications to an area, so that the area can be processed in the MUF 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 MUF that is not done through DBUTLTY.

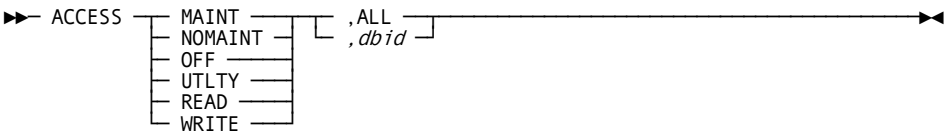
Note: The ACCESS MUF startup option has not been changed for support of area level DBUTLTY control, nor has the ACCESS console command been changed. Area level access is provided *only* through DBUTLTY.

For more information about area level DBUTLTY control, see Area Level DBUTLTY Control.

How to Use the Console Command

The MUF must be active when you 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 MUF 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). Except, note that in Simplify mode some DBUTLTY functions ignore this restriction.

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.) Except, note that in Simplify mode some DBUTLTY functions ignore this restriction.

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

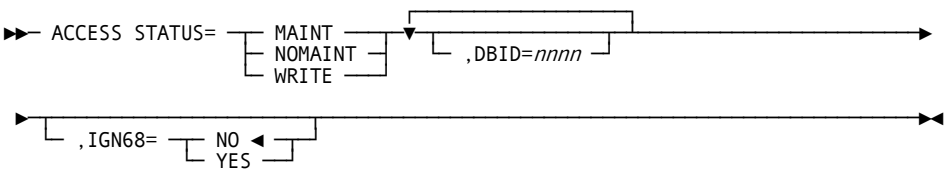
Specifies the database ID for a single database. You can specify only one per function.

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

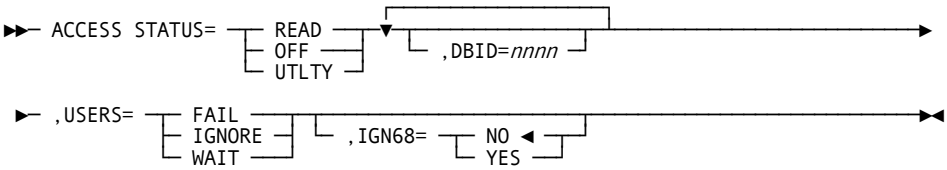
How to Use the DBUTLTY Command

The MUF must be active when you execute this command. To use the ACCESS function, execute DBUTLTY using the following:

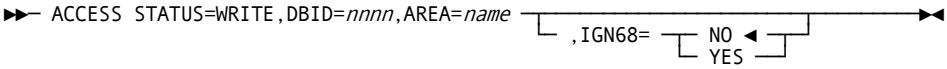
ACCESS STATUS (MAINT/NOMAINT/WRITE, Database Level)



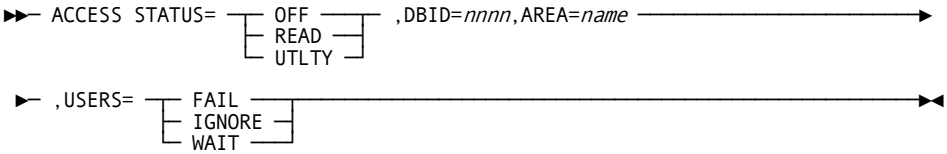
ACCESS STATUS (READ/OFF/UTLTY, Database Level)



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 about area level DBUTLTY control, see Area Level DBUTLTY Control.

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 MUF can have the area open *without* having any open User Requirements Tables when using the OPTIMIZE option of the ACCESS MUF 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 MUF can have the area open *without* having any open User Requirements Tables when using the OPTIMIZE option of the ACCESS MUF 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*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the 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

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=0M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          ACCESS   STATUS=READ,USERS=IGNORE
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

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

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.

```
                DB00607I - BASE ALL   ACCESS READ
```

This page of the report shows the following:

- A message indicating that DBUTLTY successfully placed the request on the MUF communication queue.

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

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 MUF.

Activating the Accounting Facility

To activate the Accounting Facility, perform the following steps:

1. Define the accounting database to CA Datacom Datadictionary. (The CA Datacom Datadictionary User Requirements Table must be defined to use the MUF.) For more information, see the *CA Datacom/DB Database and System Administration Guide*.
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 90) 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) and INIT IXX (Format Index Area) 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) for details.
6. Change your MUF 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 82) for details.
8. If the accounting tables were specified in the MUF startup options with a status of close, you can use them by issuing an ACCT OPEN (see following information). If not, you can add them by issuing the console command ACCT_ADD (for details, see the *CA Datacom/DB Database and System Administration Guide*), then issue the ACCT OPEN.
9. You can permanently change your MUF startup options with an ACCT statement per table including the desired status at MUF startup. This change is then used the next time MUF is reinitialized. For details, see the *CA Datacom/DB Database and System Administration Guide*.
10. If the MUF startup option ACCTPRM was specified, then temporary changes to the account tables can be done with the console-like ACCT ADD and ACCT DELETE commands during execution.

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 MUF 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 MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. For more information about XCF, see the *CA Datacom/DB Database and System Administration Guide*.

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 MUF 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

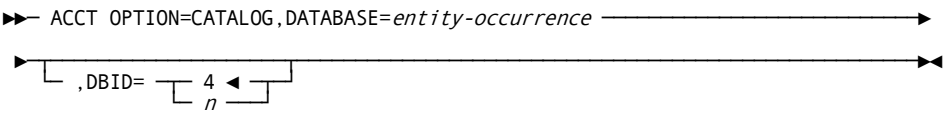
The accounting database that you are cataloging must be in PRODuction status in CA Datacom Datadictionary. If not, an error results. The CATALOG option validates the accounting database before cataloging it in the PRM table. The EDIT option of accounting also performs a validation. Run EDIT before placing a database definition in PRODuction status. See ACCT EDIT (Edit Accounting Database Definitions) (see page 90) and the *CA Datacom/DB Database and System Administration Guide* for further details.

DSF Modules

This option uses the Datadictionary Service Facility (DSF). This requires the availability of the IPC load library to this JOB/STEP. For more information about which releases of IPC are supported, see the *CA Datacom/DB Installation Guide*.

How to Use

The MUF must be active when you 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 Datadictionary.

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.

```
//jobname    See the note above and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=0M
//STEPLIB    See the note above and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN    DD *                                  Command input
         ACCT  OPTION=CATALOG,DATABASE=ACCOUNTING
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

                    FUNCTION=ACCT
                       DATABASE=ACCOUNTING
                       OPTION=CATALOG
```

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.
- If there are any messages related to syntax processing, they are displayed following the analysis of keywords.

```
                         ACCOUNTING CATALOG OPTION FOR TABLE ACCOUNTING-A00(PROD) IN DATABASE
DBID OF ACCOUNTING BASE:     5
DATABASE TABLE NAME: A00
DATABASE KEY NAME: A00KY
DATADICTIONARY KEY NAME: A00KY
    STARTING AT OFFSET 00000 FOR A LENGTH OF 012 WITH DD NAME OF A00KY
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
    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
    MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS: 11
DB13211I - ACCOUNTING DEFINITIONS FOR TABLE A00 NOW IN PRODUCTION STATUS
```

This page of the report shows the following:

- Message displaying table name or database name and version/status being cataloged.
- Accounting database ID defined to CA Datacom Datadictionary, followed by the table name and key name defined to CA Datacom Datadictionary.
- Length and offset for the specified key.

- List of the elements in the table. One line for each element, identifying each element's offset and length.
- 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 MUF 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.
- Message indicating that the definitions for this Accounting table have been placed in PROD status.

```
                          ACCOUNTING CATALOG OPTION FOR TABLE ACCOUNTING-A02(PROD) IN DATABASE
DBID OF ACCOUNTING BASE:     5
DATABASE TABLE NAME: A02
CONDITIONAL EXPRESSION:
                              (TINDX="FF")
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
```

This page of the report shows the following:

At CONDITIONAL EXPRESSION, the conditional expression that limits the Accounting data that goes to this table is displayed. 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 MUF 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 (see ACCESS (Restricting Opens or Maintenance)). 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.

MUF Connection

This function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. For more information about XCF, see the *CA Datacom/DB Database and System Administration Guide*.

When using a console command, these steps are unnecessary.

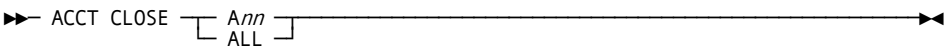
Scheduling OPENs and CLOSEs

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 95)).

How to Use the Console Command

The MUF must be active when you 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:

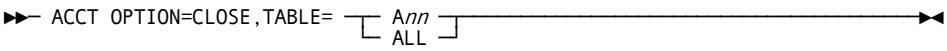
A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

How to Use the DBUTLTY Command

The MUF must be active when you 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=0M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          ACCT     OPTION=CLOSE,TABLE=ALL
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

FUNCTION=ACCT
   OPTION=CLOSE
   TABLE=ALL
```

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.
- If there are any messages related to syntax processing, they are displayed following the analysis of keywords.

```
DB13105I - ACCT               CLOSE    REQUEST ACCEPTED
```

This page of the report shows the message indicating that DBUTLTY successfully placed the request on the MUF communication queue.

ACCT EDIT (Edit Accounting Database Definitions)

The ACCT EDIT function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. For more information about XCF, see the *CA Datacom/DB Database and System Administration Guide*.

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 Datadictionary.

Database Definitions in TEST Status

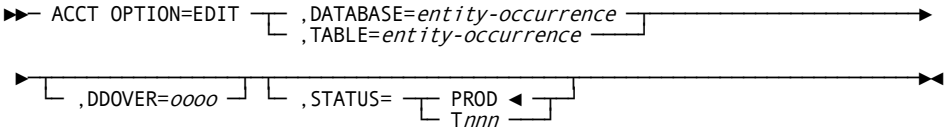
The EDIT option is recommended when building an accounting database for the first time. Since EDIT can be run against a TEST status database definition in CA Datacom Datadictionary, you can validate the structure, locating any errors before placing the definition in PRODuction status. EDIT is also recommended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

DSF Modules

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

How to Use

The MUF must be active when you 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 Datadictionary override code for the database, if it is secured with CA Datacom Datadictionary security.

Valid Entries:

A 4-character CA Datacom Datadictionary 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 Datadictionary version number in the form Tnnn.
- If the database is in PRODuction status in your CA Datacom Datadictionary, 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.

```
//jobname    See the note above and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=0M
//STEPLIB    See the note above and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN    DD *                                  Command input
         ACCT  OPTION=EDIT,DATABASE=ACCOUNTING
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

FUNCTION=ACCT
   DATABASE=ACCOUNTING
   OPTION=EDIT
```

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.

```
                      ACCOUNTING EDIT OPTION FOR TABLE ACCOUNTING-A00(PROD) IN DATABASE
DBID OF ACCOUNTING BASE:     5
DATABASE TABLE NAME: A00
DATABASE KEY NAME: A00KY
DATADICTIONARY KEY NAME: A00KY
    STARTING AT OFFSET 00000 FOR A LENGTH OF 012 WITH DD NAME OF A00KY
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
    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
    MINIMUM VALUE FOR MAXELRQ PARAMETER IN MULTI-USER STARTUP OPTIONS: 11
DB13211I - ACCOUNTING DEFINITIONS FOR TABLE A00 NOW IN PRODUCTION STATUS
```

This page of the report shows the following:

- Message displaying table name or database name and version/status being cataloged.
- Accounting database ID defined to CA Datacom Datadictionary, followed by the table name and key name defined to CA Datacom Datadictionary.
- Length and offset for the specified key.
- List of the elements in the table. One line for each element, identifying each element's offset and length.
- 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 MUF 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 MUF startup option MAXELRQ based on the number of elements actually used.

- Message indicating that the definitions for this Accounting table have been placed in PROD status.

```
                          ACCOUNTING EDIT OPTION FOR TABLE ACCOUNTING-A02(PROD) IN DATABASE
DBID OF ACCOUNTING BASE:     5
DATABASE TABLE NAME: A02
CONDITIONAL EXPRESSION:
                              (TINDX="FF")
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
```

This page of the report shows the following:

At CONDITIONAL EXPRESSION, there is a 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 MUF 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. The ACCT OPEN function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

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. It can also be used to open any tables dynamically added through the ACCT_ADD console command.

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.

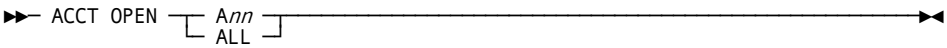
Scheduling OPENs and CLOSEs

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 87)).

How to Use the Console Command

The MUF must be active when you 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:

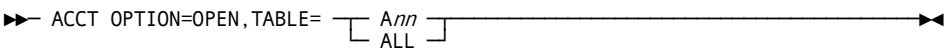
A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

How to Use the DBUTLTY Command

The MUF must be active when you 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=0M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
           ACCT     OPTION=OPEN,TABLE=A01
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

                    FUNCTION=ACCT
                       OPTION=OPEN
                       TABLE=A01
```

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.

```
DB13105I - ACCT                OPEN     REQUEST ACCEPTED
```

This page of the report shows the message indicating that DBUTLTY successfully placed the request on the MUF communication queue.

ACCT SPILL (Force Accounting Data Recording)

The ACCT SPILL function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. For more information about XCF, see the *CA Datacom/DB Database and System Administration Guide*.

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.

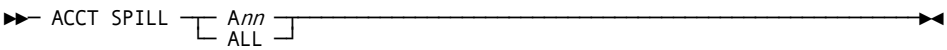
Length of Time to Process

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

The MUF must be active when you 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:

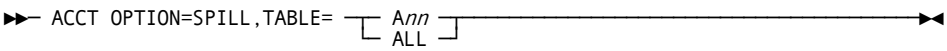
A*nn* (where *nn* is two numeric digits) or ALL

Default Value:

(No default)

How to Use the DBUTLTY Command

The MUF must be active when you 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=0M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
           ACCT     OPTION=SPILL,TABLE=A09
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

                    FUNCTION=ACCT
                       OPTION=SPILL
                       TABLE=A09
```

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.

```
DB13105I - ACCT                SPILL    REQUEST ACCEPTED
```

This page of the report shows the message indicating that DBUTLTY successfully placed the request on the MUF communication queue.

Chapter 5: AUTOCOLL (Collect and Manage Snapshots)

The AutoCollect functions record information in Snapshots of selected Dynamic System Tables. These Snapshots are recorded in a permanent database so you can run reports and analyze the results.

The SNAPSHOT function of AUTOCOLL (and AUTOSTAT) is unique in the ability to communicate to two MUFs at the same time. When the SNAPSHOT executes, it uses an internal communication path to connect to the MUF that is the source of the Snapshot. This connection requires that the Snapshot must be executed on the same LPAR as the source MUF. For the Snapshot to store its collected data, the Snapshot connects to the repository MUF using the standard DBSIDPR connection path. This connection path supports the standard pathways such as SVC, XCF, and CCI. When executing a SNAPSHOT function, be aware of the following:

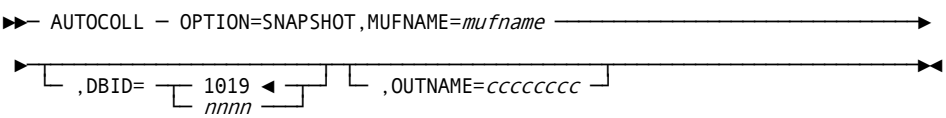
- Verify that DBUTLTY is communicating with the desired repository MUF by executing DBUTLTY with the same System Identifier module (DBSIDPR) used by the MUF.
- Verify proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.
- AutoCollect does not look at or record any information about additional buffer pools specified with the MUF startup option BUFFER_POOL_DEF. That is, AutoCollect only preserves and provides feedback for the standard buffer pools. For details about the additional buffer pools, see the *CA Datacom/DB Database and System Administration Guide*.

Create a Snapshot

Create Snapshots to record information from the selected Dynamic System Tables of the source MUF and record them in a permanent database on the repository MUF. Once Snapshots have been created, you can run additional steps to analyze and report the results.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=SNAPSHOT

Specifies to create a Snapshot rowset.

MUFNAME=

Specifies the name of the source MUF.

Valid Entries:

Valid MUF name (1 to 8 characters)

Default Value:

(No default)

Optional Keywords

DBID=

Specifies the database ID of the Snapshot database where the Snapshot will be recorded.

Valid Entries:

1 to 5000

Default Value:

1019

OUTNAME=

Specifies an alternate character value to use as the MUFNAME stored on the rows written to the Snapshot database. This allows you to organize the collected data under selected MUFNAMEs as needed. The OUTNAME option is typically used in special cases only.

Valid Entries:

1 to 8 characters

Default Value:

(No default)

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL  OPTION=SNAPSHOT,MUFNAME=MUF4
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOCOLL    OPTION=SNAPSHOT,MUFNAME=MUF4


FUNCTION=AUTOCOLL
   MUFNAME=MUF4
   OPTION=SNAPSHOT
```

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.

```
USING DST  DBID:  1000
USING SNAP DBID:  1019
MUFNAME:          MUF4
OUTNAME:          MUF4
RELEASE:          12
REPOSITORY  MUF:  MUF4

ENABLED TIME:     2008/11/10 10:53:01
CURRENT MUF TIME: 2008/11/10 10:58:21

SNAPPING TABLE: MUF_ACCOUNTING
 - RECORDS PROCESSED: 1
SNAPPING TABLE: MUF_AREA_STATS
 - RECORDS PROCESSED: 13
SNAPPING TABLE: MUF_COVEREDVIRTUAL
 - RECORDS PROCESSED: 5
SNAPPING TABLE: MUF_INTERNAL_STATS
 - RECORDS PROCESSED: 1
SNAPPING TABLE: MUF_OPTIONS
 - RECORDS PROCESSED: 1
SNAPPING TABLE: MUF_RETURN_CODES
 - RECORDS PROCESSED: 2
SNAPPING TABLE: MUF_SMP_STATS
 - RECORDS PROCESSED: 1
SNAPPING TABLE: MUF_SYSTEM_STATS
 - RECORDS PROCESSED: 1
SNAPPING TABLE: MUF_TABLE_STATS
 - RECORDS PROCESSED: 93
SNAPPING TABLE: SQL_MISC_STATS
 - RECORDS PROCESSED: 1
SNAPPING TABLE: SQL_SQLCODES
 - RECORDS PROCESSED: 1
SNAPPING TABLE: MUF_CONFIG
 - RECORDS PROCESSED: 2
SNAPPING TABLE: MUF_CBS
 - RECORDS PROCESSED: 1
SNAPPING TABLE: MUF_TCB_OR_SRB
 - RECORDS PROCESSED: 12
SNAPPING TABLE: MUF_SRB_ZIIP
 - RECORDS PROCESSED: 1

TOTAL ROWS SNAPPED:   136
```

Process a Snapshot Rowset into a Delta Rowset

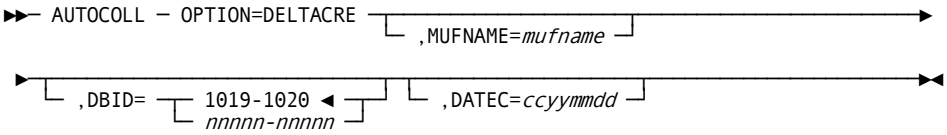
A Delta is the calculated difference between two Snapshots taken of the same MUF instance. When calculating a Delta, the beginning date and time are taken from the beginning Snapshot, while the ending date and time are taken from the ending Snapshot. The duration column is the number of hours between the beginning and ending date/times. The value of a character column or fixed value column is selected from the ending Snapshot. When calculating a Delta of a counter column, the result is the value of the ending Snapshot minus the value of the starting Snapshot. Counter values can be zero, but they cannot be a negative value.

When to Use

Use the AUTOCOLL OPTION=DELTACRE function when you want to create Delta rowsets from the Snapshot rowsets.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=DELTACRE

Specifies to create a Delta rowset from a Snapshot rowset.

Optional Keywords

MUFNAME=

Specifies the mufname of the rows on the Snapshot tables that are to be selected for processing (1 - 8 bytes).

Valid Entries:

A valid mufname contained on one or more Snapshot rowsets.

Default Value:

(No default) If this keyword is not specified, all mufnames are selected for processing.

DBID=

Specifies the database ID of the Snapshot database where the Snapshot rowsets are located and the DBID of the Delta database where the Delta rowsets will be recorded. If this parameter is specified, both DBIDs must be provided separated by a dash ("-"). You must always specify the lower DBID first.

Valid Entries:

Two 5 digit DBIDs separated by a dash ("-").

Default Value:

1019-1020 or 01019-01020

DATEC=

Specifies that only Snapshot rowsets with a MUF enable date (S_MUF_ENABLE) greater than or equal to this date will be processed into Delta rowsets. Use the format *ccyymmdd*.

Valid Entries:

8 digits

Default Value:

(No default) If this keyword is not specified, all dates will be selected for processing.

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL OPTION=DELTACRE,DATEC=20081110
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOCOLL    OPTION=DELTACRE,MUFNAME=MUF4

FUNCTION=AUTOCOLL
   MUFNAME=MUF4
   OPTION=DELTACRE
```

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.

```
TABLE MFO
DELTAS COMPUTED    : 508
DELTAS WRITTEN     : 508
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFO END

TABLE MFY
DELTAS COMPUTED    : 4,572
DELTAS WRITTEN     : 4,572
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFY END

TABLE MFA
DELTAS COMPUTED    : 174,221
DELTAS WRITTEN     : 174,221
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFA END

    .
    .
    .

TABLE BVD
DELTAS COMPUTED    : 508
DELTAS WRITTEN     : 508
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE BVD END

TABLE TOT
DELTAS COMPUTED    : 347,899
DELTAS WRITTEN     : 347,899
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE TOT END
```

User Created SUMMARY Delta Rowset

In addition to the system created Delta rowsets created by the DELTACRE function, you can create additional Delta rowsets based on the combination of existing Delta rowsets into a new rowset known as a DELTA SUMMARY rowset. Keywords are provided to allow you to select the Delta rowsets that will be included for summarization. During the summary function, special data handling is performed according to the following:

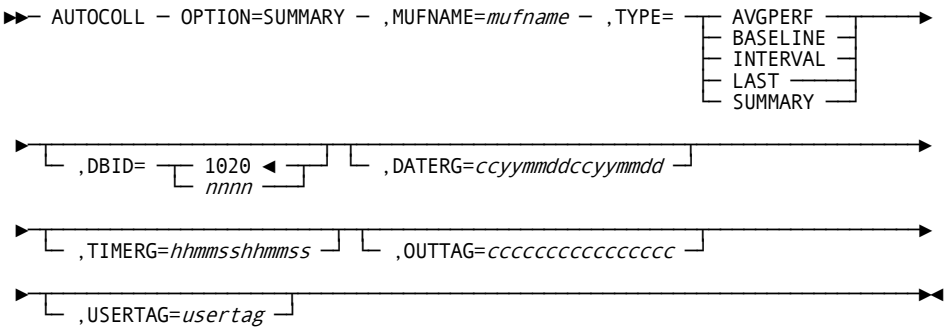
- Beginning date and time information is taken from the earliest rowset processed.
- Ending date and time information is taken from the oldest rowset processed.
- The duration column is the sum of the duration of the selected rowsets.
- Counter columns are added together (summarized).
- Non-counter columns are set to the value of the oldest (ending) rowset.

When to Use

Use the AUTOCOLL OPTION=SUMMARY when you would like to add together one or more Delta rowsets to create a new summary rowset.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=SUMMARY

Specifies to total the counter columns from all Delta rowsets selected and build a new summary rowset.

Valid Entries:

SUMMARY

Default Value:

(No default)

MUFNAME=

Only Delta rowsets with this mufname will be selected for processing.

Valid Entries:

1- to 8-byte valid mufname found on one or more of the Delta rowsets.

Default Value:

(No default)

TYPE=

Specifies the type of Delta rowsets selected as input for the processing. The type of rowset is identified in the D_MUF_TYPE column.

AVGPERF

Identifies a user created Delta rowset that was built by the AVGPERF function.

BASELINE

Identifies a user created Delta rowset that was built by the BASELINE function.

INTERVAL

Identifies a DELTA INTERVAL rowset that was built by the DELTACRE function.

LAST

Identifies a DELTA LAST rowset that was built by the DELTACRE function.

SUMMARY

Identifies a user created Delta rowset that was built by the SUMMARY function.

Valid Entries:

AVGPERF, BASELINE, INTERVAL, LAST, or SUMMARY

Default Value:

(No default)

Optional Keywords

DATERG=

Specifies the range of dates (D_MUF_END_DATETIME) for the selected rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

(No default) All dates will be selected for processing.

DBID=

Specifies the database ID of the Delta database from which the Delta rowsets are both read and written.

Valid Entries:

1 to 5000

Default Value:

1020

OUTTAG=

Places a 1- to 16-byte character tag in the USERTAG column on each Delta row produced by this execution. You can then specify that tag value as a selection criteria in future executions.

Valid Entries:

1 to 16 characters

Default Value:

(No default) If the keyword is not specified, the USERTAG column will be set to blanks.

TIMERG=

Specifies the range of times (D_MUF_END_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

(No default) All times will be selected for processing.

USERTAG=

Specifies to restrict processing to a Delta rowset that has the specified value in the USERTAG column.

Valid Entries:

1 to 16 characters

Default Value:

(No default) All USERTAG values will be selected for processing.

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL   OPTION=SUMMARY,MUFNAME=MUF4,TYPE=INTERVAL,OUTTAG=INTERVAL-SUM
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOCOLL OPTION=SUMMARY,MUFNAME=MUF4,TYPE=INTERVAL,
  DATERG=2007090120071020,TIMERG=160000170000

FUNCTION=AUTOCOLL
   DATERG=2007090120071020
   MUFNAME=MUF4
   OPTION=SUMMARY
   TIMERG=160000170000
   TYPE=INTERVAL
```

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.

```
TABLE MFO
DELTAS COMPUTED    : 1
DELTAS WRITTEN     : 1
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFO END

TABLE MFY
DELTAS COMPUTED    : 9
DELTAS WRITTEN     : 9
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFY END

TABLE MFA
DELTAS COMPUTED    : 349
DELTAS WRITTEN     : 349
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFA END
   .
   .
   .
TABLE BVD
DELTAS COMPUTED    : 1
DELTAS WRITTEN     : 1
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE BVD END

TABLE TOT
DELTAS COMPUTED    : 699
DELTAS WRITTEN     : 699
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE TOT END
```

User Created BASELINE Delta Rowset

In addition to the system created Delta rowsets created by the DELTACRE function, you can create additional Delta rowsets based on the combination of existing Delta rowsets into a new rowset known as a DELTA BASELINE rowset. A Baseline rowset is created by summarizing the selected rowsets and then dividing the duration and counter values by the number of rowsets being summarized. Keywords are provided to allow you to select the Delta rowsets that will be included for processing. During the BASELINE function, special data handling is performed according to the following:

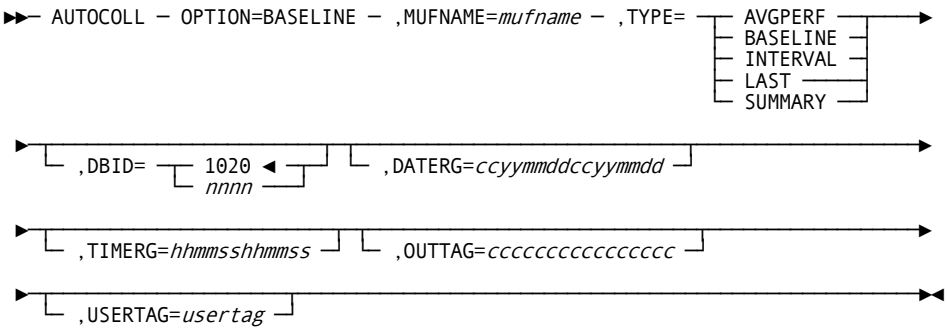
- Beginning date and time information is taken from the earliest rowset processed
- Ending date and time information is taken from the oldest rowset processed
- The duration column is the sum of the duration of the selected rowsets divided by the number of input rowsets
- Counter columns are added together (summarized) and then divided by the number of input rowsets
- Non-counter columns are set to the value of the oldest (ending) rowset

When to Use

Use the AUTOCOLL OPTION=BASELINE when you would like to add together one or more Delta rowsets and average them creating a new Baseline rowset.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=BASELINE

Specifies to total the counter columns from all Delta rowsets selected and build a new Baseline rowset.

Valid Entries:

BASELINE

Default Value:

(No default)

MUFNAME=

Only Delta rowsets with this mufname will be selected for processing.

Valid Entries:

1- to 8-byte valid mufname found on one or more of the Delta rowsets.

Default Value:

(No default)

TYPE=

Specifies the type of Delta rowsets selected as input for the processing. The type of rowset is identified in the D_MUF_TYPE column.

AVGPERF

Identifies a user created Delta rowset that was built by the AVGPERF function.

BASELINE

Identifies a user created Delta rowset that was built by the BASELINE function.

INTERVAL

Identifies a DELTA INTERVAL rowset that was built by the DELTACRE function.

LAST

Identifies a DELTA LAST rowset that was built by the DELTACRE function.

SUMMARY

Identifies a user created Delta rowset that was built by the SUMMARY function.

Valid Entries:

AVGPERF, BASELINE, INTERVAL, LAST, or SUMMARY

Default Value:

(No default)

Optional Keywords

DATERG=

Specifies the range of dates (D_MUF_END_DATETIME) for the selected rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

(No default) All dates will be selected for processing.

DBID=

Specifies the database ID of the Delta database from which the Delta rowsets are both read and written.

Valid Entries:

1 to 5000

Default Value:

1020

OUTTAG=

Places a 1- to 16-byte character tag in the USERTAG column on each Delta row produced by this execution. You can then specify the value for selection in future executions.

Valid Entries:

1 to 16 characters

Default Value:

(No default) If the keyword is not specified, the USERTAG column will be set to blanks.

TIMERG=

Specifies the range of times (D_MUF_END_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

(No default) All times will be selected for processing.

USERTAG=

Specifies to restrict processing to a Delta rowset that has the specified value in the USERTAG column.

Valid Entries:

1 to 16 characters

Default Value:

(No default) All USERTAG values will be selected for processing.

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL OPTION=BASELINE,MUFNAME=MUF4,TYPE=LAST
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOCOLL OPTION=BASELINE,MUFNAME=MUF4,TYPE=INTERVAL,
  DATERG=2007090120071020,TIMERG=160000170000

FUNCTION=AUTOCOLL
   DATERG=2007090120071020
   MUFNAME=MUF4
   OPTION=BASELINE
   TIMERG=160000170000
   TYPE=INTERVAL
```

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.

```
TABLE MFO
DELTAS COMPUTED    : 1
DELTAS WRITTEN     : 1
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFO END

TABLE MFY
DELTAS COMPUTED    : 9
DELTAS WRITTEN     : 9
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFY END

TABLE MFA
DELTAS COMPUTED    : 349
DELTAS WRITTEN     : 349
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFA END
   .
   .
   .
TABLE BVD
DELTAS COMPUTED    : 1
DELTAS WRITTEN     : 1
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE BVD END

TABLE TOT
DELTAS COMPUTED    : 699
DELTAS WRITTEN     : 699
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE TOT END
```

User Created AVGPERF Delta Rowset

In addition to the system created Delta rowsets created by the DELTACRE function, you can create additional Delta rowsets based on the combination of existing Delta rowsets into a new rowset known as a DELTA AVGPERF rowset. Keywords are provided to allow you to select the Delta rowsets that will be included for summarization. During the AVGPERF function, special data handling is performed according to the following:

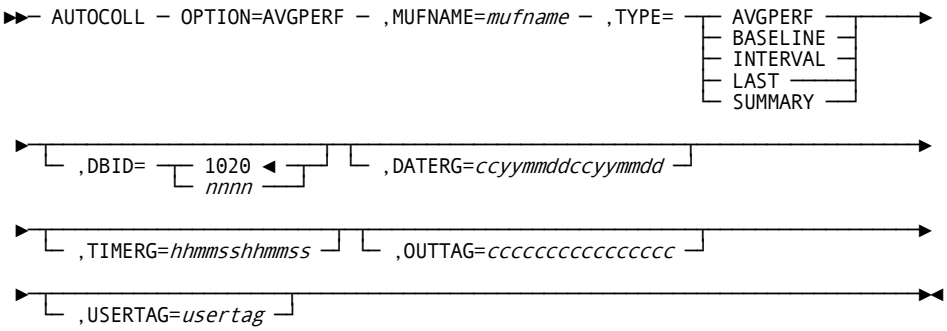
- Beginning date and time information is taken from the earliest rowset processed
- Ending date and time information is taken from the oldest rowset processed
- The duration column is set to 1.0 (1 hour)
- Counter columns are added together (summarized) and then divided by the total of all duration hours from the input rowsets
- Non-counter columns are set to the value of the oldest (ending) rowset

When to Use

Use the AUTOCOLL OPTION=AVGPERF when you would like to add together one or more Delta rowsets to create a new average performance (1 hour duration) rowset.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=AVGPERF

Specifies to total the counter columns from all Delta rowsets selected and build a new Summary rowset.

Valid Entries:

AVGPERF

Default Value:

(No default)

MUFNAME=

Only Delta rowsets with this mufname will be selected for processing.

Valid Entries:

1- to 8-byte valid mufname found on one or more of the Delta rowsets.

Default Value:

(No default)

TYPE=

Specifies the type of Delta rowsets selected as input for the processing. The type of rowset is identified in the D_MUF_TYPE column.

AVGPERF

Identifies a user created Delta rowset that was built by the AVGPERF function.

BASELINE

Identifies a user created Delta rowset that was built by the BASELINE function.

INTERVAL

Identifies a DELTA INTERVAL rowset that was built by the DELTACRE function.

LAST

Identifies a DELTA LAST rowset that was built by the DELTACRE function.

SUMMARY

Identifies a user created Delta rowset that was built by the SUMMARY function.

Valid Entries:

AVGPERF, BASELINE, INTERVAL, LAST, or SUMMARY

Default Value:

(No default)

Optional Keywords

DATERG=

Specifies the range of dates (D_MUF_END_DATETIME) for the selected rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

(No default) All dates will be selected for processing.

DBID=

Specifies the database ID of the Delta database from which the Delta rowsets are both read and written.

Valid Entries:

1 to 5000

Default Value:

1020

OUTTAG=

Places a 1- to 16-byte character tag in the USERTAG column on each Delta row produced by this execution. You can then specify the value for selection in future executions.

Valid Entries:

1 to 16 characters

Default Value:

(No default) If the keyword is not specified, the USERTAG column will be set to blanks.

TIMERG=

Specifies the range of times (D_MUF_END_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

(No default) All times will be selected for processing.

USERTAG=

Specifies to restrict processing to a Delta rowset that has the specified value in the USERTAG column.

Valid Entries:

1 to 16 characters

Default Value:

(No default) All USERTAG values will be selected for processing.

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL OPTION=AVGPERF,MUFNAME=MUF4,TYPE=LAST,OUTTAG=AVGPERF-LAST
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOCOLL OPTION=AVGPERF,MUFNAME=MUF4,TYPE=INTERVAL,
  DATERG=2008010120081231,TIMERG=160000170000,OUTTAG=AVGPERF-2008

FUNCTION=AUTOCOLL
   DATERG=2008010120081231
   MUFNAME=MUF4
   OPTION=AVGPERF
   OUTTAG=AVGPERF-2008
   TIMERG=160000170000
   TYPE=INTERVAL
```

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.

```
TABLE MFO
DELTAS COMPUTED    : 1
DELTAS WRITTEN     : 1
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFO END

TABLE MFY
DELTAS COMPUTED    : 9
DELTAS WRITTEN     : 9
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFY END

TABLE MFA
DELTAS COMPUTED    : 349
DELTAS WRITTEN     : 349
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE MFA END
   .
   .
   .
TABLE BVD
DELTAS COMPUTED    : 1
DELTAS WRITTEN     : 1
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE BVD END

TABLE TOT
DELTAS COMPUTED    : 700
DELTAS WRITTEN     : 700
DUPLICATES IGNORED : 0
DELTAS WITH ERROR  : 0
TABLE TOT END
```

Create DSV Output Datasets

You can select Delta rows based on various column values and output the rowsets content as a Delimiter Separated Value (DSV) file, then use available FTP protocols to download the sequential output files from the mainframe to the PC as text files. These TXT files can be downloaded into a PC-based spreadsheet product like Microsoft Excel.

During the OPTION=DSVOUT execution, as the utility processes each of the Delta tables, it will attempt to open a matching sequential dataset for each of the tables. The sequential dataset DDname is specified as:

OUT*xxx*

The *xxx* is the three character table name in the Delta database.

If the DDNAME is missing from the JCL, the printed report indicates that no output dataset could be opened and skips the DSV processing for that table. The utility attempts to process all 17 tables generating output for those datasets that are defined.

To allow you to place output from multiple DSVOUT functions into a single set of OUT*xxx* sequential files, the following JCL example utilizes the DISP=(MOD...) JCL parameter. MOD allows each DSVOUT function to add its selected output to the "end" of the OUT*xxx* datasets.

Note: Before running the DBUTLTY OPTION=DSVOUT function with the DISP=MOD JCL, you should remove any unwanted data in the OUT*xxx* datasets or delete them entirely.

Two Delta rowset tables that should provide immediate benefit to you for the MUF performance analysis are the Business Value Metrics (BVM) and Business Value Details (BVD) tables. These tables provide key performance measurement ratios.

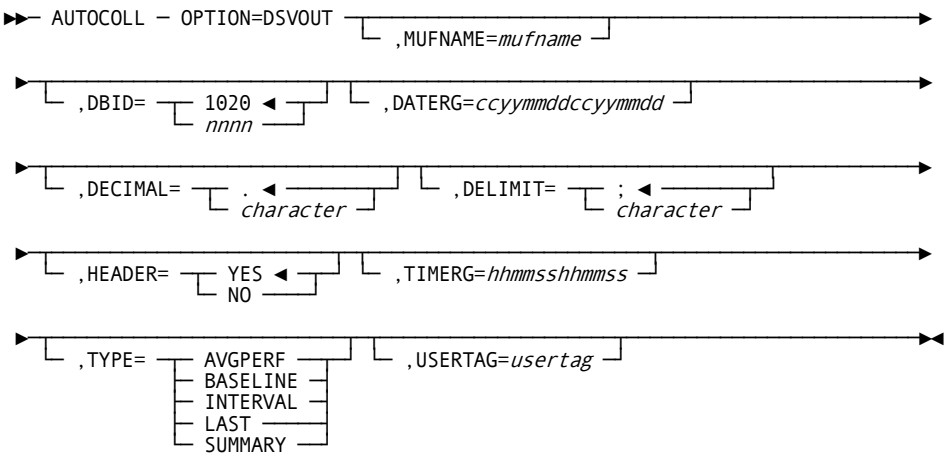
Once loaded into a spreadsheet, you can perform a variety of editing, sorting, and summarization tasks to manipulate the data into a usable form for your site.

When to Use

Use the AUTOCOLL OPTION=DSVOUT function to produce a DSV file to download to your PC.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=DSVOUT

Specifies to create a DSV output dataset.

Valid Entries:

DSVOUT

Default Value:

(No default)

Optional Keywords

MUFNAME=

Specifies the mufname of the row on the Delta tables that is to be selected to process (1 - 8 bytes).

Valid Entries:

A valid mufname contained on one or more Delta rowsets.

Default Value:

(No default) If this keyword is not specified, all mufnames will be selected for processing.

DATERG=

Specifies the range of dates (D_MUF_END_DATETIME) for the selected Delta rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

(No default) All dates will be selected for processing.

DECIMAL=

Specifies the character to use for the decimal point, such as a period (.) or a comma (,).

Valid Entries:

One character enclosed in single quotes

Default Value:

period ('.')

DELIMIT=

Specifies the character to use as the delimiter between values in the DSV output. Do not use a period (.) or a comma (,) as these special characters occur normally throughout the AutoCollect data and would cause the DSV output to be unusable.

Valid Entries:

One character

Default Value:

semicolon (;)

DBID=

Specifies the database ID of the Delta database from which the Delta rowsets are both read and written.

Valid Entries:

1 to 5000

Default Value:

1020

HEADER=

Specifies whether to produce a header row (using column names) as the first row of the DSV output.

Valid Entries:

NO or YES

Default Value:

YES

TIMERG=

Specifies the range of times (D_MUF_END_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

(No default) All times will be selected for processing.

TYPE=

Specifies to select the rows with the following in the MUFTYPE column.

AVGPERF

Select only Delta rows with a MUFTYPE of AVGPERF.

BASELINE

Select only Delta rows with a MUFTYPE of BASELINE.

INTERVAL

Select only Delta rows with a MUFTYPE of INTERVAL.

LAST

Select only Delta rows with a MUFTYPE of LAST.

SUMMARY

Select only Delta rows with a MUFTYPE of SUMMARY.

Valid Entries:

AVGPERF, BASELINE, INTERVAL, LAST, or SUMMARY

Default Value:

(No default) All TYPEs will be selected for processing.

USERTAG=

Specifies to restrict processing to a Delta rowset that has the specified value in the USERTAG column.

Valid Entries:

1 to 16 characters

Default Value:

(No default) All USERTAGs will be selected for processing.

Example JCL

Following is an example of the JCL for this 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.

```
 //jobname    See the previous note and JCL Requirements.
 //       EXEC PGM=DBUTLTY,REGION=2M
 //STEPLIB    See the previous note and JCL Requirements.

//OUTBVM   DD DSN=TEMP.DSVOUT.BVM,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTBVD   DD DSN=TEMP.DSVOUT.BVD,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMCB   DD DSN=TEMP.DSVOUT.MCB,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMCF   DD DSN=TEMP.DSVOUT.MCF,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFA   DD DSN=TEMP.DSVOUT.MFA,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFC   DD DSN=TEMP.DSVOUT.MFC,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFO   DD DSN=TEMP.DSVOUT.MFO,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFS   DD DSN=TEMP.DSVOUT.MFS,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFT   DD DSN=TEMP.DSVOUT.MFT,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFU   DD DSN=TEMP.DSVOUT.MFU,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFV   DD DSN=TEMP.DSVOUT.MFV,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMFW   DD DSN=TEMP.DSVOUT.MFW,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
```

```
//OUTMFY   DD DSN=TEMP.DSVOUT.MFY,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMTC   DD DSN=TEMP.DSVOUT.MTC,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTMZI   DD DSN=TEMP.DSVOUT.MZI,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTSQM   DD DSN=TEMP.DSVOUT.SQM,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//OUTSQQ   DD DSN=TEMP.DSVOUT.SQQ,DISP=(MOD,CATLG),
//            UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//SYSIN    DD *                                  Command input
 AUTOCOLL OPTION=DSVOUT,DELIMIT=;,DECIMAL='.',TYPE=BASELINE,
          USERTAG=BASELINE-2008,HEADER=YES
 AUTOCOLL OPTION=DSVOUT,DELIMIT=;,DECIMAL='.',TYPE=SUMMARY,HEADER=NO
 /*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Sample Report

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
AUTOCOLL OPTION=DSVOUT,DELIMIT=;,DECIMAL='.',TYPE=BASELINE,
USERTAG=BASELINE-2008,HEADER=YES

FUNCTION=AUTOCOLL
   DECIMAL=.
   DELIMIT=;
   HEADER=YES
   OPTION=DSVOUT
   TYPE=BASELINE
   USERTAG=BASELINE-2008
```

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.

```
TABLE MFO
RECORDS IN TABLE: 742
RECORDS SELECTED: 2
TABLE MFO END

TABLE MFC
RECORDS IN TABLE: 2,214
RECORDS SELECTED: 4
TABLE MFC END
   .
   .
   .
TABLE BVD
RECORDS IN TABLE: 742
RECORDS SELECTED: 4
TABLE BVD END
```

```
DB30073I - TOTAL RECORDS SELECTED: 1,404
```

Reporting Snapshot Rowsets

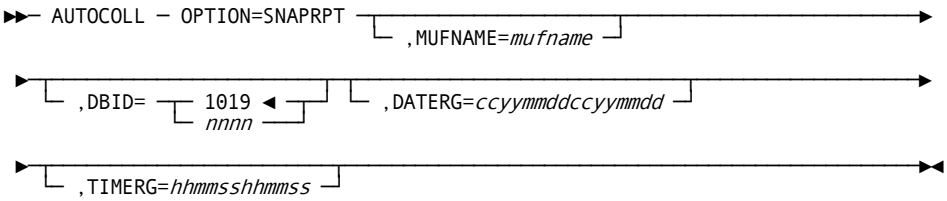
You can produce reports on specific rowsets based on your specified criteria. When you want to either create new Delta rowsets (DELTACRE) or to delete rowsets, the first step should be to produce a Snapshot report.

When to Use

Use the AUTOCOLL OPTION=SNAPRPT function when you want to produce a report of the rows.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=SNAPRPT

Specifies to create a report of Snapshot rowsets.

SNAPRPT

Specifies to list each of the Snapshot rowsets and the number of rows in each Snapshot table that belongs to that rowset.

Valid Entries:

SNAPRPT

Default Value:

(No default)

Optional Keywords

MUFNAME=

Specifies the name of the MUF (1 - 8 bytes).

Valid Entries:

A valid mufname contained on one or more of the Snapshot rowsets.

Default Value:

(No default) If this keyword is not specified, all mufnames will be selected for processing.

DATERG=

Specifies the range of dates (S_MUF_DATETIME) for the selected rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

(No default) All dates will be selected for processing.

DBID=

Specifies the database ID of the database from which the Delta rowsets are both read and written.

Valid Entries:

1 to 5000

Default Value:

1019

TIMERG=

Specifies the range of times (S_MUF_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

(No default) All times will be selected for processing.

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL OPTION=SNAPRPT,MUFNAME=MUF4
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                     AUTOCOLL    OPTION=SNAPRPT

                    FUNCTION=AUTOCOLL
                       OPTION=SNAPRPT
```

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.

```
SNAPSHOT REPORT:
================
                                                  M   M   M   M   M   M   S   S   M       M       M   M   M   M   M
                                                  F   F   F   F   F   F   Q   Q   F       F       F   C   C   T   Z
MUFNAME       MUF-ENABLED         MUF-DATE-TIME   Y   V   O   U   W   S   M   Q   C       A       T   B   F   C   I
--------  -------------------  -------------------  --- --- --- --- --- --- --- --- --- ------- ------- --- --- --- ---

MUF4      2008/08/26 05:16:12  2008/08/27 09:00:04   9   1   1  11   1   1   1   1   2     343     306   0   0   0   0
MUF4      2008/08/26 05:16:12  2008/08/27 16:15:07   9   1   1  17   1   1   1   1   2     343     306   0   0   0   0
MUF4      2008/08/26 05:16:12  2008/08/28 09:00:02   9   1   1  17   1   1   1   1   2     347     308   0   0   0   0
MUF4      2008/08/26 05:16:12  2008/08/28 16:15:25   9   1   1  18   1   1   1   1   2     347     308   0   0   0   0
MUF4      2008/08/26 05:16:12  2008/08/29 09:00:02   9   1   1  18   1   1   1   1   2     347     308   0   0   0   0
MUF4      2008/08/26 05:16:12  2008/08/29 16:15:10   9   1   1  18   1   1   1   1   2     347     308   0   0   0   0
MUF4      2008/08/26 05:16:12  2008/08/30 09:00:02   9   1   1  18   1   1   1   1   2     347     308   0   0   0   0
```

```
GRAND TOTALS:
==============

ROWSETS LISTED:     540

TOTAL ROWS:     367,311

    MFY     MFV     MFO     MFU     MFW     MFS     SQM     SQQ     MFC       MFA       MFT     MCB     MCF     MTC     MZI
------- ------- ------- ------- ------- ------- ------- ------- ------- --------- --------- ------ ------ ------ ------
  4,860     540     540   8,947     540     540     540     540   1,634   184,427   164,203      0      0      0      0
```

Reporting Delta Rowsets

You can produce reports on specific rowsets based on your specified criteria. When you want to:

- Create new Delta rowsets (SUMMARY/BASELINE/AVGPERF)
- Determine what rowsets to use in DSVOUT processing
- Delete rowsets

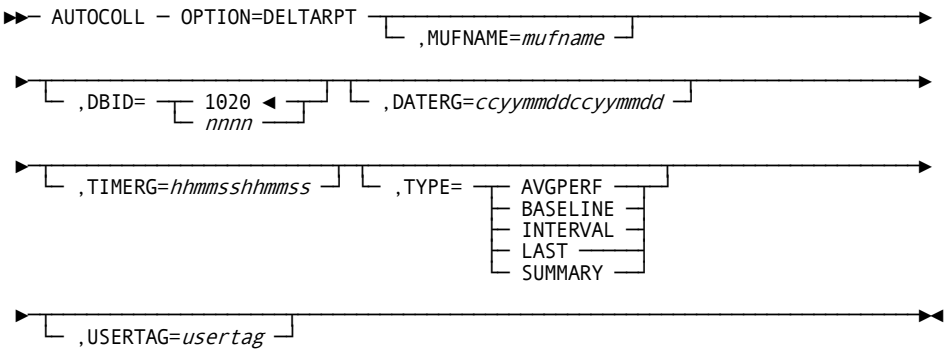
Your first step should be to produce a Delta report.

When to Use

Use the AUTOCOLL OPTION=DELTARPT function when you want to produce a report of the rows.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=DELTARPT

Specifies to create a report of Delta rowsets.

DELTARPT

Specifies to list each of the Delta rowsets and the number of rows in each Delta table that belongs to that rowset.

Valid Entries:

DELTARPT

Default Value:

(No default)

Optional Keywords

MUFNAME=

Specifies the name of the MUF (1-8 bytes).

Valid Entries:

A valid mufname contained on one or more Delta rowsets.

Default Value:

(No default) If this keyword is not specified, all mufnames will be selected for processing.

DBID=

Specifies the database ID of the Delta database from which the Delta or rowsets are both read and written.

Valid Entries:

1 to 5000

Default Value:

1020

DATERG=

Specifies the range of dates (D_MUF_END_DATETIME) for the selected rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

(No default) All dates will be selected for processing.

TIMERG=

Specifies the range of times (D_MUF_END_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

(No default) All times will be selected for processing.

TYPE=

Specifies to select the rows with the following in the MUFTYPE column.

AVGPERF

Select only rows with a MUFTYPE of AVGPERF.

BASELINE

Select only rows with a MUFTYPE of BASELINE.

INTERVAL

Select only rows with a MUFTYPE of INTERVAL.

LAST

Select only rows with a MUFTYPE of LAST.

SUMMARY

Select only rows with a MUFTYPE of SUMMARY.

Valid Entries:

AVGPERF, BASELINE, INTERVAL, LAST, or SUMMARY

Default Value:

(No default) All TYPEs will be selected for processing. Process all.

USERTAG=

Specifies to restrict processing to a Delta rowset that has the specified value in the USERTAG column.

Valid Entries:

1 to 16 characters

Default Value:

(No default) All USERTAGs will be selected for processing.

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                    Command input
AUTOCOLL OPTION=DELTARPT,MUFNAME=MUF4
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
\                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                   AUTOCOLL    OPTION=DELTARPT,MUFNAME=MUF4

                  FUNCTION=AUTOCOLL
                     MUFNAME=MUF4
                     OPTION=DELTARPT
```

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.

```
DELTA REPORT:
==============
                                                          B B   M M M  M M M S  S   M      M      M M  M   M   M
                                                          V V   F F F  F F F Q  Q   F      F      F C  C   T   Z
 MUF-START-TIME      MUF-END-TIME    TYPE     USERTAG     DURAT M D   Y V O  U W S M  Q   C      A      T B  F   C   I
_________________  _________________ ____ _____________ ______ _ _ ___ _ _ __ _ _ _ __ ___ ______ ______ _ __ ___ ___

MUFNAME: MUF4      MUF-ENABLED:   8/26/08 05:16:12
 8/26/08 05:16:12   8/27/08 09:00:04 INTV                 27.73 1 1   9 1 1 11 1 1 1  1   2    343    306 0  0   0   0
 8/27/08 09:00:04   8/27/08 16:15:07 INTV                  7.25 1 1   9 1 1 17 1 1 1  1   2    343    306 0  0   0   0
 8/27/08 16:15:07   8/28/08 09:00:02 INTV                 16.74 1 1   9 1 1 17 1 1 1  1   2    347    308 0  0   0   0
 8/28/08 09:00:02   8/28/08 16:15:25 INTV                  7.25 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0
 8/28/08 16:15:25   8/29/08 09:00:02 INTV                 16.74 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0
 8/29/08 09:00:02   8/29/08 16:15:10 INTV                  7.25 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0
 8/29/08 16:15:10   8/30/08 09:00:02 INTV                 16.74 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0
 8/30/08 09:00:02   8/30/08 16:15:19 INTV                  7.25 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0
 8/30/08 16:15:19   8/31/08 09:00:02 INTV                 16.74 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0
 8/31/08 09:00:02   8/31/08 16:15:07 INTV                  7.25 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0
 8/26/08 05:16:12   8/31/08 16:15:07 LAST                130.98 1 1   9 1 1 18 1 1 1  1   2    347    308 0  0   0   0

MUFNAME: MUF4      MUF-ENABLED:   9/02/08 03:39:24
 9/02/08 03:39:24   9/04/08 09:00:04 INTV                 53.34 1 1   9 1 1 11 1 1 1  1   2    343    306 0  0   0   0
 9/04/08 09:00:04   9/04/08 16:15:06 INTV                  7.25 1 1   9 1 1 15 1 1 1  1   2    343    306 0  0   0   0
   .
   .
   .
```

```
GRAND TOTALS:
==============

ROWSETS LISTED:     508

TOTAL ROWS:     347,899

  BVM   BVD    MFY    MFV    MFO    MFU    MFW    MFS    SQM    SQQ    MFC       MFA       MFT   MCB   MCF   MTC   MZI
_____ _____ ______ ______ ______ ______ ______ ______ ______ ______ ______ _________ _________ _____ _____ _____ _____
  508   508  4,572    508    508  8,478    508    508    508    508  1,579   174,221   154,985     0     0     0     0
```

Deleting Snapshot Rowsets

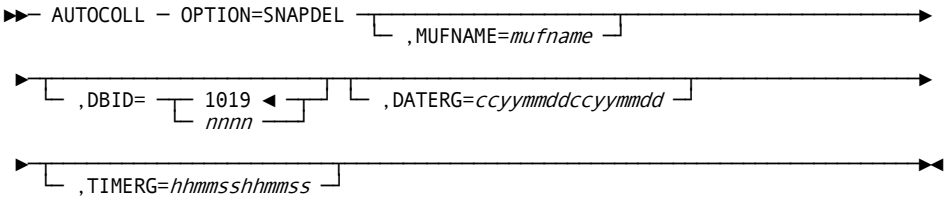
You can delete specific rowsets based on your specified criteria.

When to Use

Use the AUTOCOLL OPTION=SNAPDEL function when you want to delete a set of Snapshot rowsets.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=SNAPDEL

Specifies to delete Snapshot rowsets.

SNAPDEL

Specifies to select Snapshot rowsets to be deleted.

Valid Entries:

SNAPDEL

Default Value:

(No default)

Optional Keywords

MUFNAME=

Specifies the name of the MUF (1 - 8 bytes).

Valid Entries:

A valid mufname contained on one or more Snapshot rowsets.

Default Value:

(No default) If this keyword is not specified, all mufnames will be selected for processing.

DATERG=

Specifies the range of dates (S_MUF_DATETIME) for the selected rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

Process all dates

DBID=

Specifies the database ID of the Snapshot database from which the Snapshot rowsets are to be deleted.

Valid Entries:

1 to 5000

Default Value:

1019

TIMERG=

Specifies the range of times (S_MUF_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

Process all

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL OPTION=SNAPDEL,MUFNAME=MUF4A
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOCOLL    OPTION=SNAPDEL,MUFNAME=MUF4A


FUNCTION=AUTOCOLL
   MUFNAME=MUF4A
   OPTION=SNAPDEL
```

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.

```
TABLE MFO
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE MFO END

TABLE MFC
RECORDS IN TABLE: 65
RECORDS DELETED : 30
TABLE MFC END

TABLE MFY
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE MFY END

TABLE MFV
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE MFV END

TABLE MFW
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE MFW END

TABLE MFU
RECORDS IN TABLE: 78
RECORDS DELETED : 36
TABLE MFU END

TABLE SQQ
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE SQQ END

TABLE SQM
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE SQM END

TABLE MFS
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE MFS END

TABLE MFA
RECORDS IN TABLE: 286
RECORDS DELETED : 132
TABLE MFA END
```

```
TABLE MFT
RECORDS IN TABLE: 1,820
RECORDS DELETED : 840
TABLE MFT END

TABLE MCB
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE MCB END

TABLE MCF
RECORDS IN TABLE: 26
RECORDS DELETED : 12
TABLE MCF END

TABLE MZI
RECORDS IN TABLE: 13
RECORDS DELETED : 6
TABLE MZI END

TABLE MTC
RECORDS IN TABLE: 156
RECORDS DELETED : 72
TABLE MTC END
```

```
DB30079I - TOTAL RECORDS DELETED : 1,176
```

Deleting Delta Rowsets

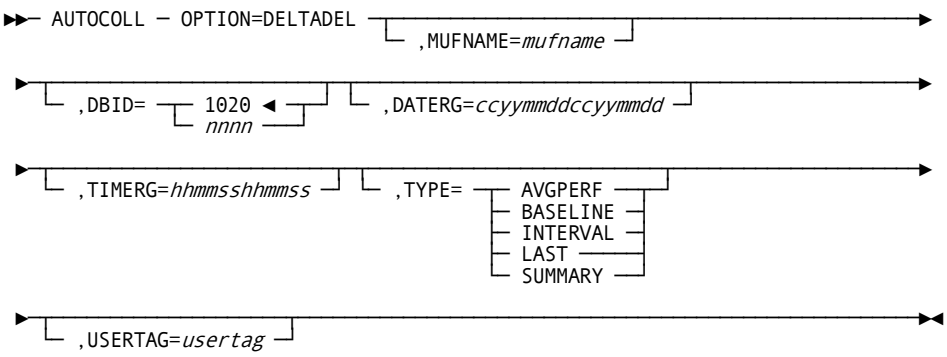
You can delete specific rowsets based on your specified criteria.

When to Use

Use the AUTOCOLL OPTION=DELTADEL function when you want to delete a set of Snapshot rowsets.

How to Use

Use the following syntax to write this command:



Command

AUTOCOLL

Invokes the AUTOCOLL function.

Required Keywords

OPTION=DELTADEL

Specifies to select Delta rowsets to be deleted.

Valid Entries:

DELTADEL

Default Value:

(No default)

Optional Keywords

MUFNAME=

Specifies the name of the MUF (1 - 8 bytes).

Valid Entries:

A valid mufname contained on one or more Delta rowsets.

Default Value:

(No default) If this keyword is not specified, all mufnames will be selected for processing.

DATERG=

Specifies the range of dates (D_MUF_END_DATETIME) for the selected rowsets to qualify for processing. Format *ccyymmddccyymmdd*

Valid Entries:

The entry must be one or more of the following:

- Starting date value is between 19000101 and 39991231
- Ending date value is between 19000101 and 39991231
- Ending date must be equal or greater than the starting date

Default Value:

Process all dates

DBID=

Specifies the database ID of the Delta database from which the Delta rowsets are to be deleted.

Valid Entries:

1 to 5000

Default Value:

1020

TIMERG=

Specifies the range of times (D_MUF_END_DATETIME) on which to base the selection. Format *hhmmsshhmmss*

Valid Entries:

The entry must be one or more of the following:

- Starting time value is between 000000 and 240000
- Ending time value is between 000000 and 240000
- Ending time must be equal or greater than the starting time

Default Value:

Process all

TYPE=

Specifies to select the rows with the following in the MUFTYPE column.

AVGPERF

Select only rows with a MUFTYPE of AVGPERF.

BASELINE

Select only rows with a MUFTYPE of BASELINE.

INTERVAL

Select only rows with a MUFTYPE of INTERVAL.

LAST

Select only rows with a MUFTYPE of LAST.

SUMMARY

Select only rows with a MUFTYPE of SUMMARY.

Valid Entries:

AVGPERF, BASELINE, INTERVAL, LAST, or SUMMARY

Default Value:

Process all

USERTAG=

Specifies to restrict processing to a Delta rowset that has the specified value in the USERTAG column.

Valid Entries:

1 to 16 characters

Default Value:

Process all

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
AUTOCOLL OPTION=DELTADEL,MUFNAME=MUF4
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOCOLL    OPTION=DELTADEL,MUFNAME=MUF4

FUNCTION=AUTOCOLL
   MUFNAME=MUF4
   OPTION=DELTADEL
```

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.

```
TABLE MFO
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE MFO END

TABLE MFC
RECORDS IN TABLE: 65
RECORDS DELETED : 25
TABLE MFC END

TABLE MFY
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE MFY END

TABLE MFV
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE MFV END

TABLE MFW
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE MFW END

TABLE MFU
RECORDS IN TABLE: 78
RECORDS DELETED : 30
TABLE MFU END

TABLE SQQ
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE SQQ END

TABLE SQM
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE SQM END

TABLE MFS
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE MFS END

TABLE MFA
RECORDS IN TABLE: 286
RECORDS DELETED : 110
TABLE MFA END
```

```
TABLE MFT
RECORDS IN TABLE: 1,820
RECORDS DELETED : 700
TABLE MFT END

TABLE MCB
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE MCB END

TABLE MCF
RECORDS IN TABLE: 25
RECORDS DELETED : 9
TABLE MCF END

TABLE MZI
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE MZI END

TABLE MTC
RECORDS IN TABLE: 145
RECORDS DELETED : 49
TABLE MTC END

TABLE BVM
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE BVM END

TABLE BVD
RECORDS IN TABLE: 13
RECORDS DELETED : 5
TABLE BVD END
```

```
DB30079I - TOTAL RECORDS DELETED : 978
```

Chapter 6: AUTOINFO (Collect Environmental MUF Information)

The AUTOINFO function collects information from memory and certain selected Dynamic System Tables and records that information into a sequential file. The collected information is also output in printed form.

Overview

The main purpose of the AUTOINFO function is to allow you to choose a given MUF and quickly generate information about that environment. This information can be helpful for you and for your communications to CA Support about a MUF problem.

AUTOINFO should be executed while the MUF is experiencing the problem. However, AUTOINFO can be executed after the MUF has come down or has been cycled. While it will not furnish as much information about a particular situation that occurred in the previous MUF execution, it still provides valuable environmental information.

If AUTOINFO is executed after a MUF has ended but has not been restarted, the function ends with a U0004 condition code. This notifies you that while the report has executed, certain data was not available.

Using AUTOINFO in DBUTLTY

The AUTOINFO function is unique in DBUTLTY because it can communicate to the MUF and its memory at the same time. When the AUTOINFO executes, it uses an internal communication path to query the memory of the MUF and uses the standard DBSIDPR connection path to gather additional data.

The AUTOINFO function communicates with one MUF. Be aware of the following:

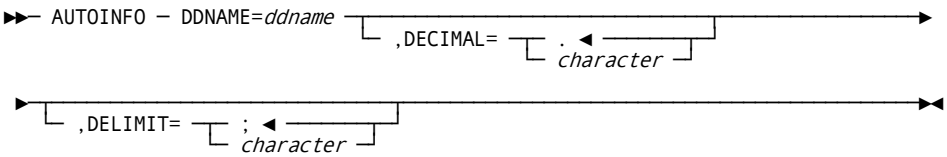
- Run the DBUTLTY on the same LPAR as the targeted MUF
- Ensure that DBUTLTY is communicating with the desired MUF by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is being used by the MUF
- Ensure proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

When to Use

Use the AUTOINFO function when you want to record information about a MUF execution, build a report, and write it as a sequential output file. CA Support may request the output file if you contact them for support.

How to Use

Use the following syntax to write the AUTOINFO command:



Command

AUTOINFO

The AUTOINFO command invokes the AUTOINFO function.

Required Keyword

DDNAME=

DDNAME= specifies the name for the diagnostic output from this execution. CA Support may request this file for analysis or help in solving an open contact.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

1 to 8 characters

Default Value:

(No default)

Optional Keywords

DECIMAL=

DECIMAL= specifies the character to use for the decimal point, such as a period (.) or a comma (,).

Valid Entries:

One character enclosed in single quotes

Default Value:

period (.)

DELIMIT=

DELIMIT= specifies the character to use as the delimiter between values in the DSV output. Do not use a period (.) or a comma (,) as these special characters occur normally throughout the AutoCollect data and would cause the DSV output to be unusable.

Valid Entries:

One character

Default Value:

semicolon (;)

Example JCL

Following is an example of the JCL for the AUTOINFO 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.

```
 //jobname    See the previous note and JCL Requirements.
 //       EXEC PGM=DBUTLTY,REGION=2M
 //STEPLIB    See the previous note and JCL Requirements.

//AUTOINFO EXEC PGM=DBUTLTY,REGION=0M
//TABLES   DD DSN=TEMP.AUTOINFO.TABLES.SUPPORT,DISP=(NEW,CATLG,CATLG),
//             UNIT=SYSDA,SPACE=(TRK,(3,1),RLSE)
//SYSPRINT DD SYSOUT=*
//SYSIN  DD  *
 AUTOINFO  DDNAME=TABLES
/*
//
```

Sample Reports

The reports produced by the AUTOINFO function look similar to the following samples. For examples of the report headers, see Sample Report Headers.

Report - MUF Available and Operational

When the MUF is available and operational, you receive a report similar to the following:

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 AUTOINFO  DDNAME=TABLES

FUNCTION=AUTOINFO
   DDNAME=TABLES
```

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.

```
MUF IS AVAILABLE, INFORMATION IS PROVIDED BY THE MUF.

MUFNAME: MUF4       VERSION: 14.0             JOB/STC NAME: QA12MUF4
CXXNAME: QAMUF4     JOBID: JOB11528              SYSTEM: CA31
ENABLED TIME:       2008/11/11 13:12:17
CURRENT MUF TIME:   2008/11/11 13:14:59

MUF IS SHADOW:      N                            MUF HAS SHADOW:    N
DB SUBSYS PRESENT:  Y                            SQL:               Y
EXTERNAL SECURITY:  N                            ACCOUNTING ON:     N

DBIDS:     DST: 1000   HISTORY: 1007   CBS TEMP: 0006   CBS HEUR: 1006
TASKS:  TOTAL: 00250      XCF: 00250       CCI: 00050      SIZE: 46080
```

```
TABLE MUF_OPTIONS: MFO

ACCTPRM_PRM_DBID        0
ACCTPRM_ANN_DBID        0
ACCTPRM_CPU_TIME        Y
...etc.
```

```
TABLE MUF_COVEREDVIRTUAL: MFC

DBID AREA V/C F/A DS   MEM_SIZE BLKS_MAX BLKS_CURR TOTAL_READS MRDF_READS VIRT_WRITES ACTV_READS NONF_READS
   6  IXX  V   N   X  262144000    64000     64000           2          2           0       NULL       NULL
  17  IXX  V   N   N      32768     NULL      NULL        NULL       NULL        NULL       NULL       NULL
  17  TTM  V   N   N   33554432     NULL      NULL        NULL       NULL        NULL       NULL       NULL
      TOT  V   N   N  295731200    64000     64000           2          2           0          0          0
      TOT  C   N   N          0        0         0           0          0           0          0          0
```

```
TABLE MUF_ACCOUNTING: MFY

DBID TABLE BUF_SIZE S THRESHOLD SPILLING TOTAL_REQU PHYSIC_IOS    LOCATES TOTAL_ADDS TOTAL_UPDT UNAVAILABL  CONFLICTS
                                          SKIPPED    ENTRIES  RECLAIMED LOCATES_ST SPIL_THRES SPIL_INTER
       TOT        0 T         0        0          0          0          0          0          0          0          0
                                                 0          0          0          0          0          0
```

```
TABLE MUF_INTERNAL_STATS: MFV

TIMES_RD_1 TIM_RD_2_8 TIM_RD_9_32 TIM_RD_33_99 TIM_WRITE_1 TIM_WRT_2_12 TIM_WRT_13_24 TIM_WRT_25_48 BRKS_DONE IO_MEM_ST
         0          0           0            0           0            0             0             0         0         0
```

```
TABLE MUF_SMP_STATS: MFW

POSTED_0                10
```

```
TABLE MUF_RETURN_CODES: MFU

EXTERNAL_CODE  INTERNAL_CODE  COUNT_CODES
TOT                                    0
```

```
TABLE MUF_SQLCODES: SQQ

SQL_CODE  CODE_COUNT
TOTAL              0
```

```
TABLE MUF_MISC_STATS: SQM

PROC_CACHE_REUSE PROC_FROM_CALL PROC_FROM_TRIGGER PROC_QUEUED PROC_SQL_STMTS PROC_FAILURES PROC_NESTS
               0              0                 0           0              0             0          0
```

```
TABLE MUF_SYSTEM_STATS: MFS

BEGIN_TIME              20081111 13.12.17.000000
CURRENT_DATETIME        20081111 13.14.59.633053
CURRENT_CPU             1
RQ_PENDING_1_10         8
RQ_PENDING_11_20        0
RQ_PENDING_21_30        0
... etc.

DATA2_USED_4            0
DATA2_USED_5            0
```

```
TABLE MUF_AREA_STATS: MFA

DBID AREA_NAME PHYSICAL_READS PHYSICAL_WRITES LOGICAL_READS LOGICAL_WRITES
   0       CXX             83              6           502              8
   0       FXX             15             14            15              0
   0       LXX              4              6             4              1
   6       IXX              0              0             4              0
1000       IXX              0              0             0              0
1000       SYS              0              0             0              0
... etc..

1006       CBS              2              0             2              0
1006       IXX              2              0             2              0
           TOT            106             26           529              9
```

```
TABLE MUF_TABLE_STATS: MFT

DBID TABLE_NAME AREA_NAME TOTAL_REQUEST TOTAL_READS TOTAL_ADDS TOTAL_DELETES TOTAL_UPDATES
1000        DRA       SYS             0           0          0             0             0
1000        DRB       SYS             0           0          0             0             0
1000        DRC       SYS             0           0          0             0             0
1000        DRD       SYS             0           0          0             0             0
1000        DRE       SYS             0           0          0             0             0
1000        DRF       SYS             0           0          0             0             0
1000        DRI       SYS             0           0          0             0             0
... etc.

1006        CBS       CBS             0           0          0             0             0
            TOT       TOT            72          56          0             0             0
```

```
TABLE MUF_CBS: MCB

BUF_CUR_PERC_FULL       0
BUF_MAX_PERC_FULL       0
CUR_USED_SET_MEM        311
MAX_USED_SET_MEM        0
SETS_DEL_MAX_AGE        16
SETS_FETCHED            0
SETS_PROCESSED          0
SETS_SPILLED            0
SETS_TEMP_INDEX         0
TEMP_INDEX_ENTRIES      0
SPILLED_JOB_NAME        NULL
SPILLED_UIB             NULL
DURATION_AT_SPILL       NULL
```

```
TABLE MUF_CONFIG: MCF

CONFIG_COMMAND             CONFIG_PARAMETERS
DIAGOPTION                 00080000000000000000000000000000000000000000000000
HISTORY                    1007....
```

```
TABLE MUF_SRB_ZIIP: MZI

SEQ.NO    ENCLAVE_SEC ZIIP_ON_CP_SEC  ZIIP_QUAL_SEC   ZIIP_SECONDS  ZIIP_PERCENT
TOTAL               0             0              0              0
```

```
TABLE MUF_TCB_OR_SRB: MTC

TASK  PROGRAM   SEQ.NO    CPU_SECONDS     TIMES_USED   TIMES_POSTED    PHYSICAL_IO
MAIN  DBMUFPR        1              0            249            248            132
 SMP  DBSMPPR        1              0              0              0              0
 SMP  DBSMPPR        2              0              0              0              0
 SMP  DBSMPPR        3              0              0              0              0
 SUB  DBCONPR        1              0              0              0              0
 SUB  DBDUSPR        1              0              0              0              0
 SUB  DBENQPR        1              0             13             13              0
 SUB  DBMSBPR        1              0              3              0              0
 SUB  DBOCSPR        1              0              5             10              0
 SUB  DBOCSPR        2              0              0              0              0
 SUB  DBOC2PR        1              0              5              5              0
 TOT              9999              0            275            276            132
```

```
TABLE BUSINESS_VALUE_METRICS: BVM

Elapsed Seconds                   162
CPU Seconds Used                  1
Physical I/O                      132

Database Requests                 27
Data Area Requests                538
Table Requests                    56
Table Reads                       56
Table Adds                        0
Table Deletes                     0
Table Updates                     0

Calculated Business Value Metrics
Requests per elapsed second       0.16
Requests per CPU second           27.00
Requests per Physical IO          0.20
```

```
TABLE BUSINESS_VALUE_METRICS: BVD

Elapsed Seconds                   162
CPU Seconds Used                  1


Request Processing                                %
Database Requests                 27              **
Data Manager Requests             30              111.1
SQL Requests                      0               0.0


Physical I/O Processing                           %
Total Physical I/O                132             **
Physical I/O - Read               106             80.3
Physical I/O - Write              26              19.6
Sequential Read Ahead I/Os        0               0.0
Accounting I/O                    0               0.0
Logging I/O                       6               4.5


Buffer Processing                                 5 to 1 Ratios
IXX Buffer Use 5+                 0               0.0
DXX Buffer Use 5+                 0               0.0
DATA Buffer Use 5+                0               0.0
DATA2 Buffer Use 5+               0               0.0

No IXX Buffer Available           0
No DXX Buffer Available           0
No DATA Buffer Available          0
No EXPAND Buffer Available        0
No TXB Buffer Available           0


MRDF Processing                                   % Saved I/O
MRDF Memory (bytes)               295731200       **
MRDF Virtual Reads                2               1.5
MRDF Virtual Writes               0               0.0

MRDF Covered Memory (bytes)       0               **
... etc.


Request Pending 31-40             0
Request Pending 41-50             0
Request Pending 50+               0
```

```
DB30073I - TOTAL RECORDS SELECTED: 101
```

Report - MUF Previously Active

When the MUF is not currently available but was previously active on the system, you receive a report similar to the following:

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                   AUTOINFO  DDNAME=TABLES

                  FUNCTION=AUTOINFO
                     DDNAME=TABLES
```

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.

```
MUF NOT AVAILABLE, INFORMATION PROVIDED FROM PRIOR EXECUTION THIS SYSTEM.

MUFNAME: MUF4       RELEASE: 12     SP:           JOB/STC NAME: QA12MUF4
CXXNAME: QAMUF4     JOBID: JOB51596               SYSTEM: CA31
ENABLED TIME:       2008/11/10 10:53:01

MUF IS SHADOW:      Y                             MUF HAS SHADOW:    N
DB SUBSYS PRESENT:  N                             SQL:               Y
EXTERNAL SECURITY:  N                             ACCOUNTING ON:     N

DBIDS:     DST: 1000   HISTORY: 1007   CBS TEMP: 0006   CBS HEUR: 1006
TASKS:  TOTAL: 00250      XCF: 00250       CCI: 00050      SIZE: 46080
```

```
DB30073I - TOTAL RECORDS SELECTED: 0

                  DB10045E - ABNORMAL TERMINATION DUE TO ERROR(S)
```

Report - MUF Not Available

When the MUF has not been started since this LPAR was last IPLed, you receive a report similar to the following:

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                   AUTOINFO  DDNAME=TABLES

                  FUNCTION=AUTOINFO
                     DDNAME=TABLES
```

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.

```
MUF NOT AVAILABLE, NO INFORMATION BECAUSE NO PRIOR EXECUTION THIS SYSTEM.
```

```
DB30073I - TOTAL RECORDS SELECTED: 0

                    DB10045E - ABNORMAL TERMINATION DUE TO ERROR(S)
```

Chapter 7: AUTOSTAT (Report on MUF Information)

The AUTOSTAT function creates a Snapshot of selected MUFs with their active tasks.

This function opens the AutoStatus database in the repository MUF using the standard DBSIDPR connection and then calls an internal API to collect information about the tasks running on one or more target MUFs in this same LPAR. The repository MUF can also be a target MUF.

To execute AutoStatus, the repository MUF must be at CA Datacom/DB Version 14.0. AutoStatus allows a target MUF to be CA Datacom/DB r11 or greater.

The AutoStatus database is installed as part of a new install or upgrade. The default database ID is 1018. If you decide to change the AutoStatus database to a different DBID, we recommend that the database be left as 1018.

The process allows a single Snapshot to collect information about up to 100 active MUFs in the targeted LPAR. Since AutoStatus uses the standard DBSIDPR to communicate to the repository MUF, the repository MUF can be on a different LPAR (assuming that you have the cross LPAR communication set-up in the repository MUF). For more information about setting up the repository MUF for cross LPAR access, see the section in the *CA Datacom/DB Database and System Administration Guide* for XCF and CCI communication.

When more than one MUF is selected as a target for an AutoStatus Snapshot, the data is collected from the target MUFs in serial fashion in alphabetic order. For AutoStatus executions that have a significant number of MUFs, the actual process may run for a few seconds. To view all of this information as a single rowset, the same timestamp is loaded into each record recorded by this Snapshot (S_MUF_DATETIME).

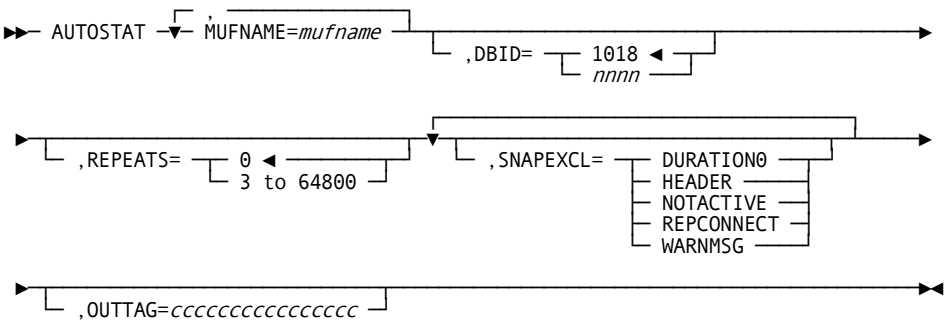
For more information about how to implement and use AutoStatus, see the *AutoScope User Guide*.

When to Use AUTOSTAT

Use the AUTOSTAT function when you want to create a Snapshot of selected MUFs with their active task information so that you can process the information and analyze the results. The data is saved in the AutoStatus database.

How to Use AUTOSTAT

Use the following syntax to write this command.



Command

AUTOSTAT

Invokes the AUTOSTAT function. Specifies to create a Snapshot of the specified MUFs with their active tasks.

Required Keyword

MUFNAME=

Identifies the MUF from which the task data is retrieved. You can repeat this parameter to include more than one MUF. Alternately, you can use a wild card character (*) to specify to select all MUFs whose names begin with the characters you specify.

Valid Entries:

The MUF name or beginning portion of a mufname with an asterisk (*)

Default Value:

(No default)

Optional Keywords

DBID=

Specifies the database ID of the database where the Snapshot rowset is recorded.

Valid Entries:

1 to 5000

Default Value:

1018

OUTTAG=

Specifies a 1- to 16-character string that is used to fill in the USERTAG column on each row in the table. You can use this in various ways. For example, if you are collecting status information from multiple LPARs, you can indicate the LPAR where a given status was collected. In another case, you can process the status rows with your own program and, once a row has been read, you could place a value in the USERTAG column to indicate it has been processed.

Valid Entries:

1 to 16 characters

Default Value:

Blanks

REPEATS=

Specifies whether the AutoStatus function should remain active and perform periodic Snapshots (until stopped by the user). A value of zero indicates that the function should run only once and then end. A numeric value between 3 and 64800 indicates that the function should remain active and the interval between the Snapshots will be set equal to the selected value in seconds. The interval is measured as the time from the end of one Snapshot and the beginning of the next Snapshot. You may want to set the parameter to a sufficient value, such as 600 (10 minutes), to keep the function from generating large numbers of rows in the AutoStatus database. If you are setting a repeats value, you need to set-up the appropriate job parameters to keep the system from canceling the job due to long execution times.

Valid Entries:

0 for none or from 3 to 64800

Default Value:

0 (Indicates that the function should not repeat)

SNAPEXCL=

Specifies the items to exclude.

DURATION0

Suppresses the creation of any detail task rows that have a DURATION of blanks. During the task inquiry, any task that has not had a given command in process for at least one second has a duration of blanks. This parameter is used to reduce the amount of detail task rows that are written to the AUTOSTAT database. With this option in place, only tasks with a command in process for a duration of one second (:01) or greater are written to the SNAPSHOT database.

HEADER

Suppresses the writing of a new page header as each SYSPRINT page fills. When you want to keep the SYSPRINT to a minimum, you can use this option.

NOTACTIVE

Suppresses the creation of any detail task rows that have a CURRENT_STATUS of NOTACTIVE. This parameter is used to reduce the amount of detail task rows that are written to the AUTOSTAT database.

REPCONNECT

Suppresses the open and close of the AUTOSTAT database URT for each Snapshot (see REPEATS). Specifying SNAPEXCL=REPCONNECT tells the utility to open the URT at the beginning of the AUTOSTAT function and to leave it open until the function is ended. In this mode, the AUTOSTAT function will issue a COMMIT at the end of each Snapshot process, but not close the URT.

Without the SNAPEXCL=REPCONNECT, the AUTOSTAT function disconnects from the repository MUF after each status Snapshot. Then, just prior to the next Snapshot, the function re-opens the URT and re-establishes communication to the repository MUF. This ensures that the URT does not remain open for long periods of time while the function is in a "wait" between Snapshots. When you want to limit the amount of opens and closes, and the associated messages in the repository MUF region, you should specify the SNAPEXCL=REPCONNECT option.

Note: With the SNAPEXCL=REPCONNECT option specified, you must terminate AUTOSTAT before issuing the EOJ to the repository MUF.

WARNMSG

Suppresses the creation of warning messages when specified MUFs are not found (not active) when the status Snapshot is executed.

Valid Entries:

DURATION0, HEADER, NOTACTIVE, REPCONNECT, WARNMSG

Default Value:

Do not exclude

Console Commands

When you specify a REPEATS= value, AUTOSTAT attaches a console subtask to allow you to communicate to AUTOSTAT while it is executing.

Any console command that changes an operational parameter except REPEATS, such as EXCLUDE DURATION0, will go into effect at the next processing cycle.

For example, if the task is running with REPEATS=600 (10 minutes), and you issue the console command EXCLUDE DURATION0, the change will be made in the operational parameters, but it will not take effect until the next Snapshot process. Issuing the REPEATS command changes the current operational value for the REPEATS parameter and also triggers an immediate Snapshot to be taken.

When you have a long REPEATS value, the console interaction allows you to make the change and also force an immediate Snapshot. In cases, where you want to force a Snapshot without changing the overall operational parameters, you can do this by just issuing the REPEATS command with a value equal to the current REPEATS value.

Issuing the AS_OPTIONS command does not change any operational values or trigger a Snapshot. AS_OPTIONS only requests that the current operational values are printed to SYSPRINT for review purposes.

Console interaction is provided through the following /STOP, /CANCEL, and /MODIFY processes.

/STOP *jobname*

Terminates the AUTOSTAT task. The process is the same as the following /MODIFY *jobname*,EOJ process.

/CANCEL *jobname*

Causes the AUTOSTAT task to be canceled with a S222 abend.

/MODIFY *jobname,command*

Issues changes using a specific command format as documented in the following section. It allows you to pass information to the active AUTOSTAT task.

MODIFY Command Format

The MODIFY command has the following format:

The jobname is the name of the AUTOSTAT job and command is the action to be taken and value is the optional value supplied to the command. There must only be a comma separating the jobname and the command and there must be a space separating the command and the value, if specified.

Console commands will be echoed to the SYSPRINT output.

Command

ADDMUF *newmuf*

Add the new MUFNAME to the list of MUFs that are being targeted for the status Snapshots. *newmuf* is 1 to 8 alphanumeric characters.

Note: There is a limit of 100 MUFNAMEs per AUTOSTAT execution.

AS_OPTIONS

Produces a listing of all AUTOSTAT options in effect to the SYSPRINT output.

COMM EOJ

Same as EOJ.

EOJ

Complete current Snapshot, set return code, and terminate the AUTOSTAT program.

EXCLUDE DURATION0

Excludes tasks that have a "blank" duration.

EXCLUDE HEADER

Tells the utility to stop producing heading pages to the SYSPRINT output file.

EXCLUDE NOTACTIVE

Excludes tasks that have a NOT ACTIVE status.

EXCLUDE REPCONNECT

Suppresses the URT open and close that occurs in conjunction with the REPEATS= parameter.

Note: With the EXCLUDE REPCONNECT option specified, you must terminate AUTOSTAT before issuing the EOJ to the repository MUF.

EXCLUDE WARNMSG

Suppresses warning messages.

INCLUDE DURATION0

Includes tasks regardless of duration.

INCLUDE HEADER

Produces a page header at the top of each new page of the SYSPRINT output file. Each page is typically 60 lines.

INCLUDE NOTACTIVE

Includes tasks regardless of task status.

INCLUDE REPCONNECT

Activates the URT open and close that occurs in conjunction with the REPEATS= parameter.

INCLUDE WARNMSG

Enables warning messages.

OUTTAG *tag1*

Changes the current OUTTAG value to a new value. Value must be between 1 and 16 alphanumeric characters.

REMOVEMUF *oldmuf*

Remove the existing MUFNAME from the list of MUFs that are being used as the source for the status Snapshots.

Note: There must be at least one valid MUFNAME per AUTOSTAT execution. If you attempt to remove the last MUFNAME, the command will be ignored and a message issued.

REPEATS *nnnnn*

Changes the current REPEATS= parameter timer to this new value. Value must be 0 or between 3 and 64800. Changing the value to 0 causes the utility to do one last Snapshot and end. Changing the value to other than 0 forces an immediate Snapshot and changes the wait interval between further Snapshots and the utility continues to execute in the loop with the new REPEATS value.

Example JCL

Following is an example of the JCL for this 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.

```
//jobname    See the previous note.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note.
//SYSIN    DD *                                  Command input
AUTOSTAT MUFNAME=DBDVMV1,MUFNAME=PRGMUF01,OUTTAG=SAMPLE
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6
                   AUTOSTAT DBID=1018,OUTTAG=OUTTAG1,
                     REPEATS=6,
                     MUFNAME=MUF4

                  FUNCTION=AUTOSTAT
                     DBID=01018
                     MUFNAME=MUF4
                     OUTTAG=OUTTAG1
                     REPEATS=00006
```

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.

```
--------------------------------------------------------------------------------
REPOSITORY DBID:  1018
REPOSITORY  MUF:  MUF4
CURRENT DATETIME: 22:58:01.6078100000 2009/03/02

PROCESSED MUF: MUF4     < MUF4     RECORDS PROCESSED: 1

TOTAL MUFS PROCESSED: 1 OUT OF 1
TOTAL ROWS SNAPPED:   1
--------------------------------------------------------------------------------
REPOSITORY DBID:  1018
REPOSITORY  MUF:  MUF4
CURRENT DATETIME: 22:58:07.6399570000 2009/03/02

PROCESSED MUF: MUF4     < MUF4     RECORDS PROCESSED: 1

TOTAL MUFS PROCESSED: 1 OUT OF 1
TOTAL ROWS SNAPPED:   1
--------------------------------------------------------------------------------
   .
   .
   .

--------------------------------------------------------------------------------
REPOSITORY DBID:  1018
REPOSITORY  MUF:  MUF4
CURRENT DATETIME: 22:58:31.7494190000 2009/03/02

PROCESSED MUF: MUF4     < MUF4     RECORDS PROCESSED: 1

TOTAL MUFS PROCESSED: 1 OUT OF 1
TOTAL ROWS SNAPPED:   1
--------------------------------------------------------------------------------
** 22:58:36.4478340000 2009/03/02
DB30050I - COMM EOJ ISSUED - PROGRAM TERMINATED
```

Chapter 8: BACKUP (Create Backups)

Making backups of critical data is essential in a data processing installation. DBUTLTY provides functions to create backups of 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.

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. When using Native Key sequence you can do a *parallel backup.* See the *CA Datacom/DB Database and System Administration Guide* for details on reorganizing a data area or loading a multiple table area.

More Information:

- For information about DBUTLTY STATUS console command support for this function, see COMM STATUS (Print Status of MUF Jobs) (see page 283).
- For more information about the impact of table partitioning on the BACKUP function, see the BACKUP Utility (see page 44).

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 (see CXXMAINT CONVERT1214 (Upgrade CXX to Current Version) for details).
- As the first step in moving the Directory to a larger extent.

Backup Output

The BACKUP function uses a variable record format and a block size equal to 4K. Do not code any DCB parameters in your JCL. 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. For more information about the SQL Object Consistency Analyzer, see the *CA Datacom/DB SQL User Guide*.

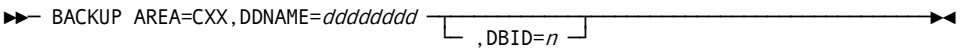
Note: You can use the SQL Object Consistency Analyzer to determine whether CA Datacom Datadictionary, Data Definition Directory, Schema Information Tables areas, and the Directory are synchronized and to determine the steps needed to correct any problems.

How to Use

You can execute this command in either Single User or with the MUF active. In Simplify mode, however, to provide current accurate information and to control (prevent) inappropriate executions, MUF is always intended to run enabled. If MUF is not enabled, it is a best practice to allow it to enable using the DELAY68= option of the DBSIDPR module. For details about the DBSIDPR module, see the *CA Datacom/DB Database and System Administration Guide*. If you know that the MUF is not enabled, and if you do not intend to enable it, and if you must force the execution of this BACKUP, declare the MUF is not to be used either by performing the BACKUP after another function that executes only with MUF down, or by following a function to establish that environment using SET OPTION1=MUF_NOT_ENABLED, which disables any DELAY68= and forces execution with MUF not enabled.

To back up your Directory, execute DBUTLTY using the command:

Back Up Directory (z/OS)



Command

BACKUP

Invokes the BACKUP function.

Required Keywords

AREA=CXX

Identifies the area to be backed up as the Directory.

,DDNAME=

Specifies the z/OS JCL DDname for the output data set. This name must match the corresponding name in your JCL.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to z/OS 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. 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)

Example JCL

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.

```
//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
//OUT3     DD DSN=cxx.bkup,DISP=(,CATLG),        Output data
//                    UNIT=tape
//SYSIN              DD *                        Command Input
         BACKUP   AREA=CXX,DDNAME=OUT3
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
BACKUP AREA=CXX,DDNAME=Z

FUNCTION=BACKUP
   AREA=CXX
   DDNAME=Z
```

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.

```
BASE PROCESSED - 1
BASE PROCESSED - 2
BASE PROCESSED - 3
BASE PROCESSED - 4
BASE PROCESSED - 5
BASE PROCESSED - 6
BASE PROCESSED - 10
BASE PROCESSED - 15
BASE PROCESSED - 16
BASE PROCESSED - 17
BASE PROCESSED - 97
BASE PROCESSED - 197
BASE PROCESSED - 297
BASE PROCESSED - 312
BASE PROCESSED - 397
BASE PROCESSED - 497
BASE PROCESSED - 597
BASE PROCESSED - 697
BASE PROCESSED - 797
BASE PROCESSED - 897
BASE PROCESSED - 990
BASE PROCESSED - 997
BASE PROCESSED - 1,000
BASE PROCESSED - 1,006
BASE PROCESSED - 1,007
BASE PROCESSED - 1,018
BASE PROCESSED - 1,019
BASE PROCESSED - 1,020
BASE PROCESSED - 1,397
BASE PROCESSED - 1,497
BASE PROCESSED - 2,009
BASE PROCESSED - 5,000
*                                    REQUEST COMPLETE
```

This page of the report shows what the CXX BACKUP backed up and gives 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 MUF 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.

In Simplify mode, the function requires MUF to be enabled. This provides control and protection. If MUF is not enabled, this function should wait for MUF to enable. If you are required to execute the function without MUF enabled, for example when you need to allow restart to execute, the function has to be made to follow a function that can only run with MUF not enabled, or it must follow a function to acknowledge MUF is not enabled with SET OPTION1=MUF_NOT_ENABLED. It is possible (but not recommended) to use SET OPTION1=MUF_ENABLED_OR_DISABLED.

A BACKUP without MULTUSE=YES and without SEQ=PHYSICAL requires the database to be closed. If the database is open with no users, it is automatically closed. A BACKUP with SEQ=PHYSICAL automatically causes a COMM STATS to the database before execution, to record current statistics and control information to the CXX and data area control block. It also forces the pipeline of write pendings for the data area.

When to Use

Use the BACKUP data area option to perform the following:

- 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 Datadictionary 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.

When using Native Key sequence you can do a parallel backup.

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 (see page 591) (Rebuild a Database) 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 MUF 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. For more information, see LOCK (Lock a Data Area) and UNLOCK.

This procedure assumes only one MUF instance is enabled when the lock is issued. If there is more than one MUF instance that can open the area for update, lock each MUF intance. *No* protection is provided if a MUF 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 MUF 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 MUF.
- The database may *not* be open for update in another MUF 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 MUF if more than one is executing.

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

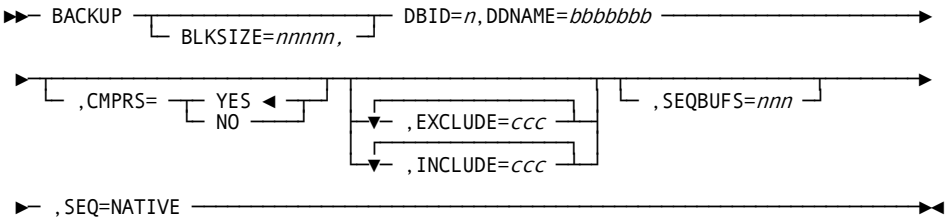
- The database will be opened by MUF for update.
- The database will have no *unloading* status set.
- The utility sets ACCESS area READ (utility set).
- The utility sets BACKUP executing this MUF this database area.

How to Use

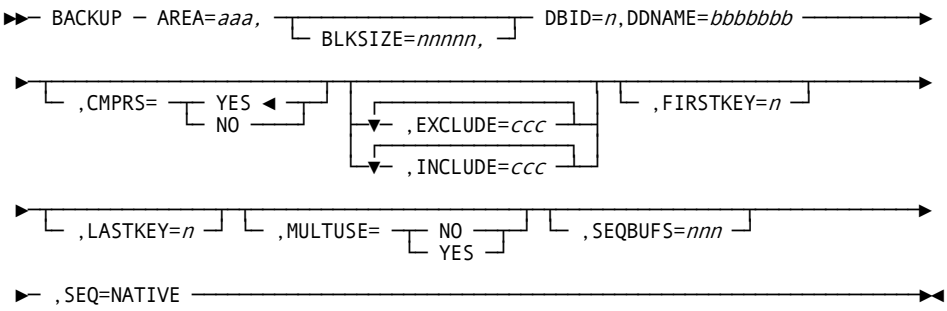
You can execute this command in either Single User or with the MUF active. In Simplify mode, however, to provide current accurate information and to control (prevent) inappropriate executions, MUF is always intended to run enabled. If MUF is not enabled, it is a best practice to allow it to enable using the DELAY68= option of the DBSIDPR module. For details about the DBSIDPR module, see the *CA Datacom/DB Database and System Administration Guide*. If you know that the MUF is not enabled, and if you do not intend to enable it, and if you must force the execution of this BACKUP, declare the MUF is not to be used either by performing the BACKUP after another function that executes only with MUF down, or by following a function to establish that environment using SET OPTION1=MUF_NOT_ENABLED, which disables any DELAY68= and forces execution with MUF not enabled.

To create a backup of a data area, execute DBUTLTY using the following command. You can code multiple EXCLUDE= or INCLUDE= keywords.

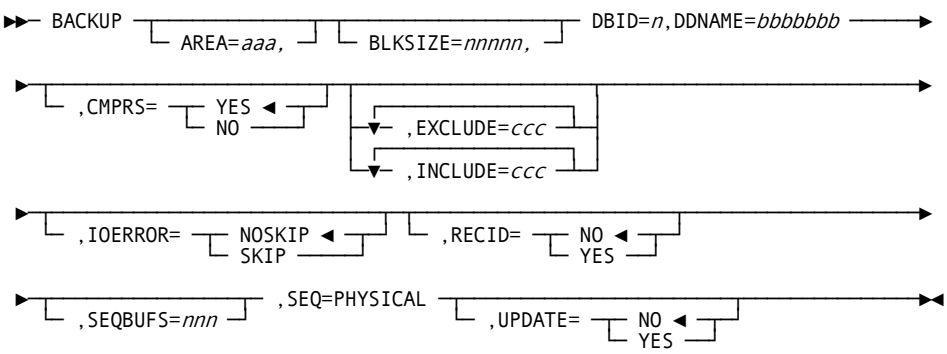
Back Up All Areas in Database Native (z/OS)



Back Up Single Area in Database Native (z/OS)



Back Up Data Area Physical (z/OS)



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)

,DDNAME=

Specifies the JCL DDname for the output data set. This name must match the corresponding name in your JCL.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to 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.

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 a default as specified in the following.

Note: If the block size is not large enough to contain the largest record in the area (see formula below), 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 = (A/128) + 1, if the data is compressed.

Valid Entries:

A 1- to 5-digit number

Default Value:

The default for BLKSIZE= when running z/OS is 0 (zero), which defaults to the z/OS default. That default is large and optimized for the particular device type, that is, the most optimal default for sites executing this operating system.

,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 abends with a user 4 abend code. 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.

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.

For more information about area level DBUTLTY control, see Area Level DBUTLTY Control.

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 Datadictionary 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

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 179).

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. One of the uses of FIRSTKEY=/LASTKEY= involves doing parallel backups.

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=

With MULTUSE=YES specified, if necessary to ensure best performance DBUTLTY overrides the SEQBUFS= value either specified or defaulted. If SEQBUFS= is omitted (invoking the default value) or specified as a value 0-128 (even numbers only), the MUF treats this as though 128 was specified (or defaulted). This directs the MUF to allocate 128 private sequential buffers for use by this function. If SEQBUFS= is specified as a number 130-256 (even numbers only) the MUF treats this as though 256 was specified and uses the common data pools (64 blocks per I/O).

When MULTUSE= is omitted or specified as NO, SEQBUFS= is edited and ignored, and DBUTLTY executes with the best performance by using either the available data buffers (specified with the DATANO= parameter in the DBMSTLST macro) or available private sequential buffers.

Valid Entries:

0 to 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 - 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.

```
//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
//OUT1     DD DSN=pay.bkup,DISP=(,CATLG),         Output data
//            UNIT=tape
//SYSIN    DD *                                   Command Input
          BACKUP    SEQ=NATIVE,AREA=DEM,DBID=1,DDNAME=OUT1
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                CONTROL CARD(S)
                .........1.........2.........3.........4.........5.........6.........7.........8
                BACKUP AREA=DEM,DBID=1,DDNAME=OUT1

                FUNCTION=BACKUP
                   AREA=DEM
                   DBID=1
                   DDNAME=OUT1
```

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.

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

This page of the report shows the following:

- 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.
- 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. Some examples with CA Datacom Presspack follow:
  - PRESSPAK RDTC01
  - PRESSPAK WEAK
  - PRESSPAK DCTF03
  - PRESSPAK DCTRNA
  - PRESSPAK STRONG

Example JCL - Physical Sequence

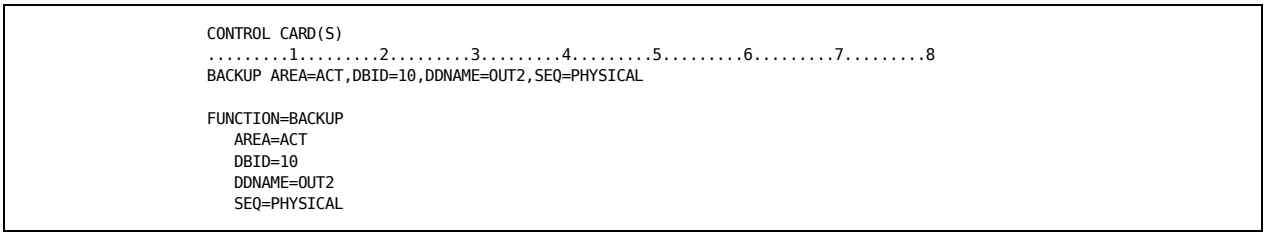
The following shows the command to back up the data area ACT in database 010 to the output data set, OUT2. Output the data in physical sequence and in compressed form (default).

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.

```
//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
//OUT2     DD DSN=act.bkup,DISP=(,CATLG),         Output data
//           UNIT=tape
//SYSIN    DD *                                   Command Input
           BACKUP   AREA=ACT,DBID=10,DDNAME=OUT2,SEQ=PHYSICAL
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

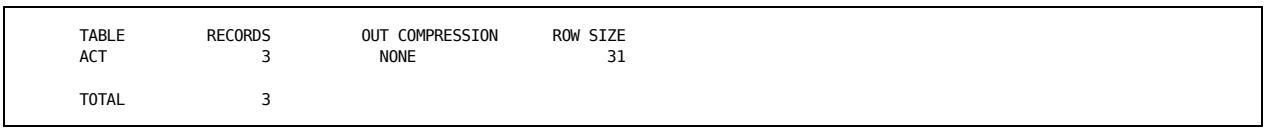


```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
BACKUP AREA=ACT,DBID=10,DDNAME=OUT2,SEQ=PHYSICAL

FUNCTION=BACKUP
   AREA=ACT
   DBID=10
   DDNAME=OUT2
   SEQ=PHYSICAL
```

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.



```
TABLE         RECORDS          OUT COMPRESSION      ROW SIZE
ACT                 3            NONE                    31

TOTAL               3
```

This page of the report shows the following:

- 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.
- 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. Some examples with CA Datacom Presspack follow:
  - PRESSPAK RDTC01
  - PRESSPAK WEAK
  - PRESSPAK DCTF03
  - PRESSPAK DCTRNA
  - PRESSPAK STRONG

Chapter 9: COMM (Communicate with MUF)

Use the COMM function to communicate a request directly to the MUF. It executes similar to a batch program through a task area in the MUF except the SNAPCSA and SNAPSVC subfunctions. 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 CA Support)
- Clear the Statistics and Diagnostics Area
- End MUF 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 a dump of the Extended Common Storage Area (ECSA)
- Print a dump of the SVC
- Print/display the status of MUF 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: Except for SNAPCSA and SNAPSVC, the COMM options require that the MUF is active.

Length of Time to Process a Request

The utility waits until the request is complete to continue.

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.

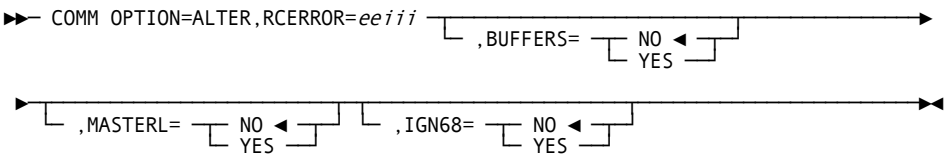
When to Use

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

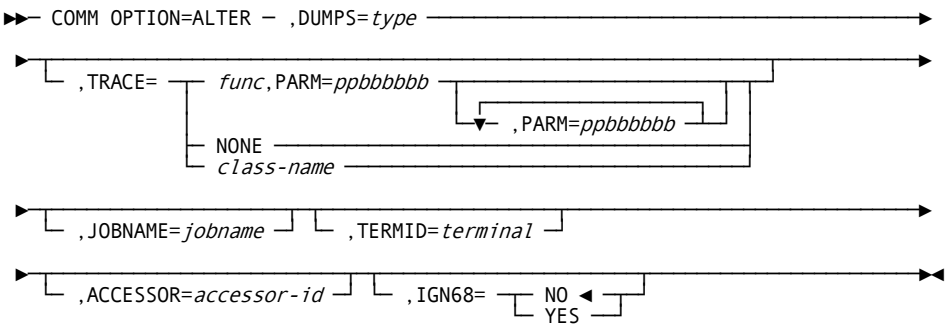
How to Use

The MUF must be active when you execute this command. Execute the COMM function with one of the following command formats:

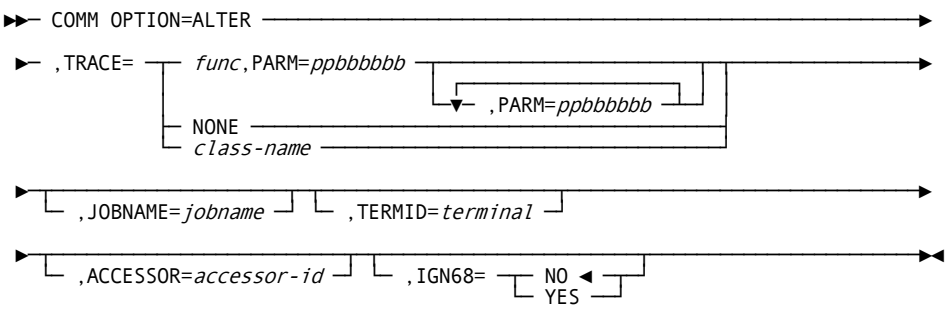
Modify Error Table, RCERROR=



Modify Error Table, DUMPS=



Modify Error Table, TRACE=



Command

COMM

Invokes the function to pass communication to the MUF.

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*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the 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

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 MUF startup option, you can specify values for *ee* and *iii.*

Note: 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:

The *pp* is 01 to 99 and specifies the position in the parameter list of the parameter to be dumped.

The *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.

CA 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 765). 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 Datadictionary transactions issued to process SQL requests

TRACEDETAIL - Causes more information to be traced for some other trace classes. See the note that follows the explanation about the MSG option. Also see the note marked with one asterisk (*).

TRACEERR - Calls to MUF

TRACEEXPR - Expression and data type

TRACEGLOBAL - Summary report showing SQL subsystem state, issued when run unit ends

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 765).)

TRACEPARSE - SQL parser

TRACEPLAN - PLAN locking

PROCTRIGPLS - Diagnostic information related to procedures and triggers. PROCTRIGPLS does *not* have a prefix of TRACE.

TRACERAAT - Record-at-a-time calls issued to process SQL requests.

Default Value:

(No default)

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
```

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
   COMM OPTION=ALTER,RCERROR=05034,MASTERL=YES,BUFFERS=YES
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

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

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.

```
DB13105I - COMM     ALTER    REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF 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 databases to be closed.

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

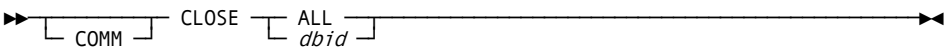
COMM CLOSE used for databases is typically used as an early step in a job stream for reorganization.

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

The MUF must be active when you 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).

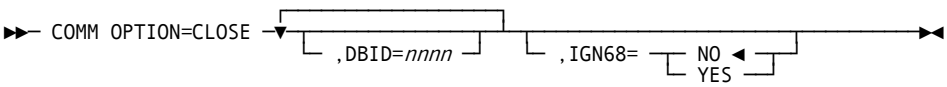
dbid

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).

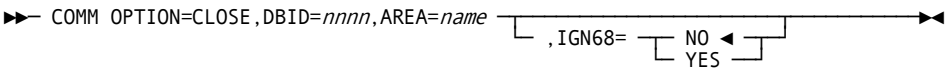
How to Use the DBUTLTY Command

The MUF must be active when you 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 MUF.

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 about area level DBUTLTY control, see Area Level DBUTLTY Control.

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*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the 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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
         COMM      OPTION=CLOSE,DBID=1
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

FUNCTION=COMM
   DBID=00001
   OPTION=CLOSE
```

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.

```
DB13105I - COMM     CLOSE    REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF communication queue.

COMM CLRnnn (Reset Alternate Output Data Sets)

In z/OS, when running MUF with a SYSOUT MUF startup option specified, you can use CLRML, CLRCBS, CLRSQL, CLRDD, and CLRDST (DBUTLTY COMM functions or console commands) to reset a PXXML, PXXCBS, PXXSQL, PXXDD, or PXXDST data set that has become full during processing.

Note: If you do not want diagnostic and tracing information routed to the Statistics and Diagnostics Area (PXX), you can use the SYSOUT MUF startup option to specify alternate output processing. The SYSOUT parameters define output for Master List snaps, CBS traces, SQL traces, CA Datacom/DB CA Datacom Datadictionary traces, and DST traces. See the *CA Datacom/DB Database and System Administration Guide* for details.

In response to a CLRML, CLRCBS, CLRSQL, CLRDD, or CLRDST, the specified SYSOUT data set is reset, that is, closed, deallocated, and reallocated (in processing the reset function, the current PXX data set is deallocated and assigned to the class specified in the SYSOUT MUF startup option). Each of the five can generate a report.

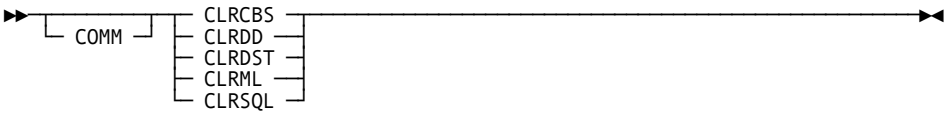
When to Use

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

How to Use the Console Command

The MUF must be active when you 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.

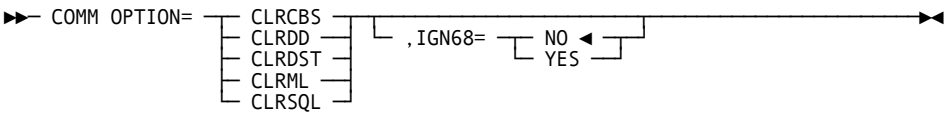
Required Keywords

CLRnnn

Requests that the specified alternate data set be reset.

How to Use the DBUTLTY Command

The MUF must be active when you execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keyword

OPTION=CLRnnn

Requests that the specified alternate data set be reset.

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.

Note: 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 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 reset the PXXML alternate data 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.

```
 //jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
 //STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
           COMM OPTION=CLRML
 /*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

                    FUNCTION=COMM
                       OPTION=CLRML
```

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.

```
DB13105I - COMM     CLRML    REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF 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 MUF. 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 MUF and are not cleared nor reset in memory.

The COMM CLRPXX function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. For more information about XCF, see the *CA Datacom/DB Database and System Administration Guide*.

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.

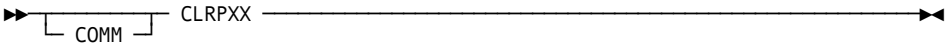
Reporting PXX Data

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.

How to Use the Console Command

The MUF must be active when you 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.

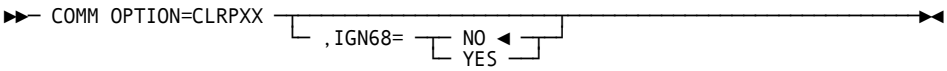
Required Keywords

CLRPXX

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

How to Use the DBUTLTY Command

The MUF must be active when you execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the MUF.

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.

Note: 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 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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          COMM     OPTION=CLRPXX
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

FUNCTION=COMM
   OPTION=CLRPXX
```

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.

```
DB13105I - COMM     CLRPXX   REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF 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 MUF.

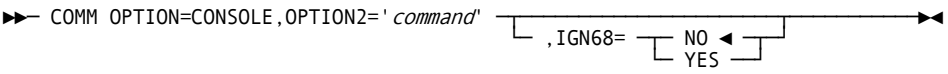
Note: For more information about console-like commands, see the section about maintaining CA Datacom/DB using console commands in the *CA Datacom/DB Database and System Administration Guide*.

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 authority to issue console commands in z/OS 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

The MUF must be active when you execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keyword

OPTION=CONSOLE

Requests that the value for *command* specified with the OPTION2= keyword be sent to the MUF.

,OPTION2='*command*'

Used to specify the command that is to be sent to the MUF. DBUTLTY passes the character string provided by OPTION2= to the MUF 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.

Note: 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*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the 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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          COMM     OPTION=CONSOLE,OPTION2='command'
/*
```

COMM EOJ (End MUF Processing)

The Comm EOJ function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. For more information about XCF, see the *CA Datacom/DB Database and System Administration Guide*.

When using a console command, these steps are unnecessary.

When to Use

Use the COMM EOJ option to end the MUF in an orderly fashion. This is the recommended way to terminate the MUF. The MUF supports being terminated as an orderly EOJ by supporting the z/OS command STOP ‘P’.

Processing of EOJ Requests

The COMM EOJ option informs the MUF that processing is to end. After receiving an EOJ request, it is acknowledged with message DB00247I - EOJ REQUESTED, NO NEW CONNECTIONS, EOJOFF AVAILABLE, and the MUF does not allow any new jobs to start processing. MUF allows jobs in progress to continue. The MUF goes to EOJ when the last job completes. A job is considered complete when all open User Requirements Tables have been closed.

There are exceptions to the closing of User Requirements Tables (URT) to support special situations. There is a URT option on the DBURSTR Macro named EOJ_OK that can be set to YES for a Job connected to MUF having a single task area. The details of the keyword are documented with the URT description. If the requirements are met, as part of EOJ detection the Job can be set as missing and the task area be set to close its URT, thereby moving the EOJ to closer execution. A second ability exists for Job connections that use multiple task areas known as EOJ_OK_S, (see the information about Extended Options Entry Processing in the C*A Datacom/DB Database and System Administration Guide*).

It is at the point of the URT closings that the EOJ is committed, noted with message: DB00243I - EOJ COMMITTED, EOJOFF DISABLED.

When the MUF is terminated an EOJ report is written to the .when 'CS7' insert SYSPRINT file. The report contains information about the following:

- An echo of MUF startup options
- A set summary information as shown in EOJ Reports.

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

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

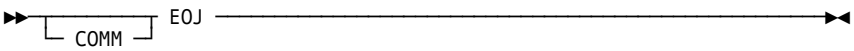
MVS STOP

The MVS STOP console command is honored by the MUF as equal to the COMM EOJ command.

How to Use the Console Command

The MUF must be active when you 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

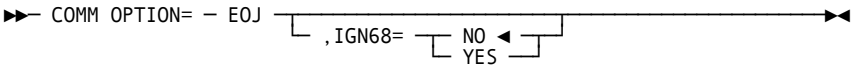
You can omit the COMM function name when submitting this console command.

EOJ

Specifies the COMM EOJ option is the desired action.

How to Use the DBUTLTY Command

The MUF must be active when you execute this command. Execute the COMM function with the following command format.



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keyword

OPTION=EOJ

Requests normal termination of the MUF.

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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          COMM     OPTION=EOJ
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

FUNCTION=COMM
   OPTION=EOJ
```

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.

```
DB13105I - COMM     EOJ      REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF communication queue.

MUF Startup Options Echoed

When the MUF is terminated, a list of the MUF startup options specified is written to SYSPRINT.

For information about the MUF startup options, see the *CA Datacom/DB Database and System Administration Guide*.

```
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 - *FLEXPOOL 2,2,2
DB01900I - *IOTASK  2                        IOTASK MAX/CURR,CURR,DSP
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
```

This page of the report shows an echo of the MUF startup options.

Summary Reports Produced by the MUF During its EOJ Process

This section shows examples of the following:

- MUF Startup Options Echoed
- Memory Resident Data Facility Summary Information
- XCF Summary Information
- MUFplex Summary Information
- Accounting Facility Summary Information
- TCB Use Summary Information

- SMP Informational Report
- Miscellaneous Diagnostic Information Report

See EOJ Reports 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

Memory Resident Data Facility Summary Information

When the MUF is terminated, summary Memory Resident Data Facility (MRDF) information is written to the SYSPRINT file after the MUF startup options.

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

```
                                M R D F     SUMMARY INFORMATION



                                                          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
```

This page of the report shows the following information:

AREA

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

USE

Indicates whether the resident area is COVERED or VIRTUAL.

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 MUF 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.

BLOCKS

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

SPACE

ADDRESS indicates no use of a data space and the data is allocated in normal memory (above the 24-bit line). DATA indicates use of a data space to contain the blocks provided. 64-BIT indicates that the data is allocated in z/OS 64-bit.

MRDF READS

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

VIRTUAL MRDF WRITES

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

COVERED 1ST/ACT READS

Provides the number of blocks read from disk for either a FIRST or ACTIVE COVERED area.

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.

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 MUF and not disabled, an Accounting report follows the MRDF report.

```
                                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
```

This page of the report shows the following information:

ADDS

Displays the number of new records added to the Accounting table.

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

CORE SIZE

Displays the buffer parameter specified for this table.

DATABASE

Displays Database ID containing the Accounting table.

ENTRIES RECLAIMED

Displays 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

Displays the 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

Displays the number of entries in the Accounting table that had exclusive control held by a user application when a spill was initiated.

EXCPS

Displays the 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 MUF startup option ACCT being exceeded.
- The value of the buffer parameter of the ACCT option being too small.

INTERNALLY INITIATED SPILLS

Displays the 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

Displays the 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**).

LOCATES INITIATED

Displays the 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 CA Support.

NUMBER OF ENTRIES

Displays the number of entries in the in-memory accounting buffer (always odd).

REQUESTS PROCESSED

Displays the number of times this Accounting table was used. This value is determined by the conditional expression, if there is one.

SKIPPED DUE TO EXCLUSIVE

Displays the number of entries that were not immediately processed due to exclusive control conflicts.

SPILLS INITIATED TO AXXTHLD

Displays the number of times that the threshold parameter value of the MUF 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

Displays the name of the Accounting table.

UPDATES

Displays the 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 of these.

XCF Summary Information

If any XCF requests are processed by the MUF, in the EOJ Report following the Memory Resident Data Facility Summary Information Report, the XCF Summary Information Report is generated. Following is an example of a XCF Summary Information Report:

```
                                    XCF SUMMARY INFORMATION

JOB/SYS NAME    GROUP         TOTAL JOBS   TOTAL REQUESTS

*               DCMDEV               2          12,014
*               DCMD2                4          37,363

BUFFER SIZE    IB-REQUESTS     OB-REQUESTS

     1K            43,268          12,014
     2K             6,107          18,209
     8K                 0           9,925
    16K                 0           9,227
```

This page of the report shows the following information:

JOB/SYS NAME

Is based on the XCFFROM startup parameter.

GROUP

Is based on the XCFFROM startup parameter.

TOTAL JOBS

Displays the count of jobs associated with this XCFFROM origin.

TOTAL REQUESTS

Displays the total requests processed for this XCFFROM origin.

BUFFER SIZE

Displays the XCF message size. This includes standard CA Datacom request fields, plus CA Datacom XCF overhead (variable), plus IBM XCF overhead (68 bytes). You can use this distribution to allocate XCF buffer sizes.

IB-REQUESTS

Displays the count of application requests for the specified buffer size.

OB-REQUESTS

Displays the count of results returned for the specified buffer size.

MUFplex Summary Information

If the MUF was an active MUFplex member, the MUFplex Summary Information Report follows the XCF Summary Information Report. Following is an example of a MUFplex Summary Information Report:

```
                                MUFPLEX SUMMARY INFORMATION

EVENT                          COUNT

ALL CALLS                        167
LIST CALLS                       126
LOCK CALLS                        14
SRB USES                         205
MESSAGES OUT                       5
MESSAGES IN                        8
LIST SETS                         38
LIST RESETS                       15
LIST READS                        33
LIST WRITES                       98
LOCK OBTAINS                       7
LOCK RELEASES                      7
MAX STR SIZE                   4,096
REQ STR SIZE                   4,096
STR NBR ENTRIES                1,714
STR NBR ELEMENT               26,476
MAX NBR ENTRIES                   16
MAX NBR ELEMENT                  211
MAX NBR BUFF                       3
```

This page of the report shows the EVENT column that lists the following statistics:

ALL CALLS

Displays the number of times DBXESPR was called.

LIST CALLS

Displays the number of times an explicit LIST request (read/write/lock set/lock reset) was made.

LOCK CALLS

Displays the number of LOCK requests that were made (obtain/release/alter).

SRB USES

Displays the number of invocations of MUFplex service SRBs by system.

MESSAGES OUT

Displays the number of MUFplex messages sent to other members of the MUFplex.

MESSAGES IN

Displays gives the number of MUFplex messages received from other members of the MUFplex.

LIST SETS

Displays the number of LIST LOCK SET requests.

LIST RESETS

Displays the number of LIST LOCK RESET requests.

LIST READS

Displays the number of LIST READ requests, explicit or implicit, using SRB.

LIST WRITES

Displays the number of LIST WRITE requests, explicit or implicit, using SRB.

LOCK OBTAINS

Displays the number of LOCK OBTAIN explicit requests.

LOCK RELEASE

Displays the number of LOCK RELEASE explicit requests.

MAX STR SIZE

Displays the maximum possible size of the Coupling Facility structure for this MUFplex (4K blocks).

REQ STR SIZE

Displays the requested size of the Coupling Facility structure for this MUFplex (4K blocks).

STR NBR ENTRIES

Displays the approximate capacity in number of LIST entries for this structure.

STR NBR ELEMENTS

Displays the approximate capacity in number of 256-byte elements for this structure.

MAX NBR ENTRIES

Displays the high-water mark for number of LIST entries simultaneously in the Coupling Facility for the MUFplex.

MAX NBR ELEMENT

Displays the high-water mark for number of 256-byte elements simultaneously in the Coupling Facility for the MUFplex.

MAX NBR BUFF

Displays the high-waiter mark for number of buffer LIST entries simultaneously in the Coupling Facility for the MUFplex.

TCB/SRB Use Summary Information

If the MUF startup option SMPTASK was specified and TCB mode was specified, this report on CPU time used by each Task Control Block (TCB) can be useful.

If the MUF startup option SMPTASK was specified and SRB mode was specified, an ENCLAVE SRB USE SUMMARY INFORMATION section occurs (see Enclave SRB Use Summary Information).

In TCB mode, the SMP subtask has a program name of DBSMPPR in the report, as it did in prior releases.

In SRB mode, the SMP task has a program name of *SRB-SMP in the report.

DBSMAPR subtasks are listed in the report with a program name of DBSMAPR.

TCB/SRB USE SUMMARY INFORMATION

ITEM	PROGRAM	USED COUNT	CPU TIME	TIMES POSTED	I/O COUNT
1	DBMUFPR	196	000:00:01.09	168	185
2	DBSMPPR	0	000:00:00.00	0	0
3	DBSMPPR	0	000:00:00.00	0	0
4	DBMSBPR	12	000:00:00.00	1	0
5	DBOCSPR	14	000:00:00.10	29	0
6	DBOCSPR	0	000:00:00.00	1	0
7	DBOC2PR	31	000:00:00.07	32	0
8	DBSPSPR	0	000:00:00.00	1	7
9	DBCONPR	1	000:00:00.00	1	0
10	DBENQPR	31	000:00:00.00	31	0
11	DBIOSPR	7	000:00:00.00	4	2
12	DBIOSPR	1	000:00:00.00	2	0
13	DBIOSPR	1	000:00:00.00	2	0
14	DBIOSPR	1	000:00:00.00	2	0

This page of the report provides the following information:

ITEM

The first item represents the MUF main task. The second and any following items represent either TCBs for subtasks or SRBs used as SMP tasks.

PROGRAM

Displays the name of the program.

DBCONPR

Console driven subtask

DBCPMPR

The Change Data Capture Monitor subtask in the source MUF (see Change Data Capture in the *CA Datacom/DB Database and System Administration Guide*.)

DBDUSPR

DBDUSPR exists, if needed, for PXX and (or) SYSOUT processing.

DBENQPR

Enqueue locking subtask

DBIOSPR

asynchronous I/O functions subtask (See MUF startup option IOTASK in *CA Datacom/DB Database and System Administration Guide*.)

DBMUFPR

Main task

DBMSBPR

Timer control subtask

DBOCSPR

Open/close subtask (In z/OS there are two, in z/VSE there is one.)

DBOC2PR

Second open/close process subtask (z/OS only)

DBSMAPR

DBSMAPR exists for each SMPTASK, if the SRB option was specified.

DBSMPPR

SMP subtask (See MUF startup option SUBTASK in *CA Datacom/DB Database and System Administration Guide*.)

DBSPSPR

Active recovery spilling subtask

USED COUNT

Lists the number of times an internal unit of work was performed. The count of internal units of work cannot be compared to any other CA Datacom/DB counter.

CPU TIME

Displays the total CPU time for the TCB or SRB used in hours, minutes, seconds, and hundredths of a second.

TIMES POSTED

Displays the number of times individual TCBs or SRBs went into a wait (no work to do) and were posted to start processing. Once any of the TCBs are posted, they run (and take new work) until no work exists to be done. The information provided in this field could be useful in tuning the ready-to-run option on the SMPTASK MUF startup option.

I/O COUNT

Displays the I/O count.

Procedure TCB Use Summary Information

Following is an example of a new report on Task Control Block (TCB) use.

```
                                PROCEDURE TCB USE SUMMARY INFORMATION

        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
```

This page of the report provides the following information:

TCB

Displays 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.

LAST PROGRAM

Displays the name of 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.

PROCEDURE STARTS

Displays the number of times this particular TCB was selected to process a user procedure (caused by a call procedure or a trigger).

TCB CPU TIME

Displays the hours, minutes, seconds, and sub-seconds of execution time.

SQL STATEMENTS

Displays 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

```
                                PROCEDURE SUMMARY INFORMATION

                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
```

This page of the report provides the following information:

PROGRAM

Displays the name of the program that was executed.

PROCEDURE STARTS

Displays the number of times this program was started for execution.

TCB CPU TIME

Displays the time this procedure spent executing.

SQL STATEMENTS

Displays the number of SQL statements executed by all occurrences of this program.

TCB FAILURES

Displays either 'LE' was unable to fetch the module or the module abended.

Enclave SRB Use Summary Information

When the SMPTASK MUF startup option has been used to specify SRB mode, the TCB/SRB USE SUMMARY INFORMATION report has a section that gives summary information about enclave SRB use.

Note: The appearance of a MUF EOJ report you receive in actual practice cannot necessarily be expected to conform exactly to the following sample.

```
                                ENCLAVE SRB USE SUMMARY INFORMATION

                    USER LIMIT  ZIIP QUALIFIED         ZIIP     ZIIP ON CP
ITEM                  PERCENT         CPU TIME     CPU TIME       CPU TIME

  1                       100    000:00:19.30   000:00:18.71   000:00:00.63
  2                       100    000:00:02.48   000:00:02.38   000:00:00.09
  3                        50    000:00:00.65   000:00:00.63   000:00:00.02
  4                        50    000:00:00.00   000:00:00.00   000:00:00.00
  5                         0    000:00:00.00   000:00:00.00   000:00:00.00
```

This page of the report provides the following information:

ITEM

Corresponds to the SMPTASK number for this SRB.

USER LIMIT PERCENT

Displays the value set at startup for this SRB either by allowing it to default to 100 percent, or by using the optional tuning MUF startup option ZIIP_USER_LIMIT (see the information about MUF startup options in the *CA Datacom/DB Database and System Administration Guide*).

ZIIP QUALIFIED CPU TIME

Displays the CPU time that has executed as qualified to run on a zIIP engine.

ZIIP CPU TIME

Displays the time work executed on the zIIP engine.

ZIIP ON CP CPU TIME

Displays the time that work was executed on CP engine because a zIIP was not available at that instant.

Note: Even when there is no user limit, not all SRB execution is qualified to run on a zIIP engine.

The times reported are provided by the operating system and might not always appear to be *in sync* with each other.

SMP Informational Report

If the MUF startup option SMPTASK was specified, the following report can be useful for some tuning and in dealing with CA Support. Similar to the SMP dynamic system table, it provides information about statistics for the entire MUF run. This report is available to SQL and non-SQL sites. Most of the information reported reflects counts of SMP waiting conditions, and these fields cannot be changed by tuning.

Each POSTED ENTRY item reflects the depth counts of a POSTED LIST that provides information about how many tasks were ready to run at various times. Periodically, the posted list is scanned and the counter at the appropriate depth is incremented by one. This is not done often and the counts are not comparable to other counts. As a group, the counts provide a clue as to whether running in SMP mode would help or hurt, the number of SMPTASKs that might be selected, and the ready-to-run count. This information can be used to tune the ready-to-run option of the SMPTASK MUF startup option.

Note: Items in the report which have a zero count are skipped.

```
                                    SMP INFORMATIONAL REPORT
EVENT                     COUNT

DXX BUFFER                  266
DATA BUFFER                  17
LOG BUFFER                   15
LOG LRU                       3
LOCK LIST                   111
OPEN TABLE                    1
TASK AREA                   126
WAIT LIST                   863
MEMORY MANAGER               23
PXX CONSOLE                   1
POSTED ENTRY  0           8,029
POSTED ENTRY  1           7,243
POSTED ENTRY  2           3,368
POSTED ENTRY  3           3,649
POSTED ENTRY  4           4,048
POSTED ENTRY  5           4,348
POSTED ENTRY  6           4,811
POSTED ENTRY  7           5,299
POSTED ENTRY  8           5,735
POSTED ENTRY  9           6,448
POSTED ENTRY 10           7,594
POSTED ENTRY 11           9,368
POSTED ENTRY 12          10,870
POSTED ENTRY 13          10,815
POSTED ENTRY 14          10,013
POSTED ENTRY 15           9,230
POSTED ENTRY 16           8,559
POSTED ENTRY 17           7,829
POSTED ENTRY 18           7,184
POSTED ENTRY 19           6,427
POSTED ENTRY 20           5,803
POSTED ENTRY 21           5,636
POSTED ENTRY 22           5,482
POSTED ENTRY 23          36,928
```

This page of the report shows the following:

Each count represents two tasks conflicting on the resource. All of these SMP locks are of short duration. Two tasks searching a particular index or data buffer for a read request get share locks and have no contention. If one task might need to update the block, it must run alone. This is true of all the locks as appropriate. All locks are global in nature, that is, only one exists except as follows:

- Data space locks exist for each data area.
- An Accounting table (ACCT TABLE) lock exists for each Accounting table.
- A TASK AREA lock exists for each RWTSA (task).
- Covered/Virtual control (MRDF CNTRL) exists for each covered or virtual area.

The following explain the event names that can appear in the report:

ACCOUNTING ENT

Incrementing counters in a particular entry in the Accounting buffer.

ACCOUNTING STS

Accounting system changes across all Accounting tables.

ACCOUNTING TBL

Accounting changes to each specific Accounting table.

ADD DELETE FLEX

Control during add or delete of a sequential or flexpool buffer.

CBS BUF SPILL

CBS buffer spill to IXX.

CBS BUFFER

Control during get/free memory in the CBS buffer.

CHECK INTERLOCK

Control during interlock detection.

COMIT WAIT BAD

Occurs when using the X_LOG_DELAY MUF startup option which enables the log COMIT delay enhancement. This is a count of times a task doing a transaction-ending command delayed for another task to write the LXX block, but none did within the one second time limit.

COMIT WAIT GOOD

Occurs when using the X_LOG_DELAY MUF startup option which enables the log COMIT delay enhancement. This is a count of times a task doing a transaction ending command delayed for another task to write the LXX block and another did, thus saving a possible LXX I/O.

COVERED VIRTUAL

Control of a covered or virtual area during block fetch/store.

DATA BUFFER

A specific data block in memory.

DATA HASH

Hash list of data blocks in memory.

DATA LRU

The Least Recently Used data chain to read a block not in memory.

DATA SPACE BUFF

Blocks in data area with free space in memory.

DATA SPACE MGMT

Data area space management.

DATA WRITE PEND

Write pending chain (pipeline) for data.

DEFRAG

A DEFRAG SMP lock is used to interact with backwards processing (for example, the LOCBR command). DEFRAG obtains an exclusive SMP lock at the start of each key ID processing. If there are no backward requests for this key ID during the life of the DEFRAG, the lock is simply maintained. If there is a backward request during the process, DEFRAG notes where it is in the DXX (the "lowest" level of the IXX that contains key values and pointers to actual rows) and frees the SMP lock to allow the backward process to run. DEFRAG then reestablishes the lock and rereads its blocks and continues as before. Backwards processing obtains a shared SMP lock.

DXX BUFFER

A specific DXX index block in memory.

DXX HASH

Hash list of DXX index blocks in memory.

DXX LRU

The Least Recently Used DXX index chain to read a block not in memory.

DXX WRITE PEND

Write pending chain (pipeline) for index.

EXPAND BUFFER

Control to get or free an expand buffer.

INDEX QUEUE

Control during get/free of entry in index queue.

IXX BUFFER

A specific IXX index block in memory.

IXX HASH

Hash list of IXX index blocks in memory.

IXX LRU

The Least Recently Used IXX index chain to read a block not in memory.

LOCK LIST

Control to add, delete, or search the primary exclusive control table.

LOG BUFFER

A specific log block in memory.

LOG LRU

The Least Recently Used data chain to read a log block not in memory (by transaction backout).

LOG VAR WAITSMP

A task is waiting on access to the Log Area (LXX) control information. This particular condition can occur with all LXX formats, but it occurs more frequently with variable logging. The counts are not within control of the user and should not be a concern, regardless of the count, that is to say, the count is for informational purposes only.

MEMORY MANAGER

Memory manager control.

MISCELLANEOUS

Non-SQL miscellaneous other locking.

OPEN TABLE

Share lock to search open table list or additions, exclusive lock to delete entries.

POSTED ENTRY

Depth count of a POSTED LIST.

POSTED LIST

Control of the posted list as required.

PXX CONSOLE

Print file (PXX) and console control.

SECURITY

Waits for SMP synchronization with external security.

SQL ATTACH

SQL control of attached tasks.

SQL GLOBAL

SQL global locking requirements.

SQL MISC

SQL miscellaneous control requirements.

SQL PLAN

SQL control during starting/stopping a plan.

SQL STATUS

SQL locking of a task status change.

SQL TTM

Control requirements to get/free a TTM block.

SQL URT

SQL control during URT open/close.

STAR

STAR storage control.

STAR TASK

STAR control task usage.

TASK AREA

Control during change in status of task area.

TSN

Control to start or stop a new transaction, or to see if a transaction is active.

USER COMPRESS

Lock during user compression routine(s).

WAIT LIST

Control of the wait list of tasks.

Five of the locks allow a diagnostic startup option (DIAGOPTION) to accumulate the time the task waited. If this has been selected (because of a request by CA Support), the following counts can print. The count reflects the total number of seconds of waiting that occurred.

WTIM MEM-MGR

Time spent waiting for the memory manager lock.

WTIM PXX

Time spent waiting for the print file (PXX) lock.

WTIM TASK AREA

Time spent waiting for the RWTSA (task) lock.

WTIM TIMER

Time spent waiting for the timer lock.

WTIM WAIT LIST

Time spent waiting for the wait list lock.

Miscellaneous Diagnostic Information Report

The Miscellaneous Diagnostic Information Report can be useful in dealing with CA Support and in some tuning. Similar to the Internal Statistics dynamic system table, it provides statistics for the entire MUF 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 needs 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 MUF 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 descriptions that follow the sample.

Note: Items in the report which have a zero count are skipped.

```
                                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
```

This page of the report shows the following:

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 MUF 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. 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 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 a DXX I/O that was already in process.

IO WAIT DATA BFR

Count of number of times a task waited on a 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 MUF 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.

Note: For information about the MUF 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.

Note: For information about the MUF 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.

Return Code Summary Information

The following example of an EOJ Report provides Return Code Summary Information:

```
                                RETURN CODE SUMMARY INFORMATION

     RETURN CODE    OCCURRENCES    SHORT TEXT

      01 (032)        3    INVALID DB COMMAND
      03 (027)        1    KEY NAME NOT IN CXX
      05 (034)        5    TABLE HAS NOT BEEN OPENED
```

This page of the report provides the following information:

RETURN CODE

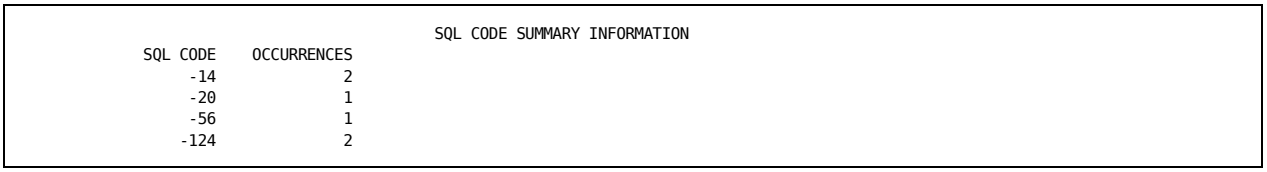
Note that these are CA Datacom/DB return codes. Not all return codes, however, are listed in these counts. Several return codes that occur frequently in non-error situations are not counted. These include (but are not restricted to) return codes: 14, 19, and 04(170). Also not counted are certain return codes that occur outside the MUF. 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.

SHORT TEXT

Displays text for the return code provided to speed the review of errors and save time that could be spent looking up message meanings.

SQL Code Summary Information



SQL CODE SUMMARY INFORMATION

SQL CODE	OCCURRENCES
-14	2
-20	1
-56	1
-124	2

This page of the report provides the following information:

SQL CODE

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 MUF 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.

```
                                MEMORY USE SUMMARY INFORMATION

DESC  LONG DESCRIPTION        24BIT-----ANY---64BIT (M=MEG,K=1024,ELSE BYTES) / OCCURRENCES

      PROGRAM MEMORY (AT EOJ)-   58-K   408-K
      DBMEMPR CALLS-----------   61       0
      ERROR, NON-SQL----------    1       0

MAIN  DBMEMPR OWNED MEMORY, 24  128-K     0
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  INDEX BFRS -------------    0       0       81-K
MRDF  COVERED/VIRTUAL 64BIT       0       0      200-K
```

This page of the report provides the following information:

DESC

Displays a short description that can be blanks and can be duplicated, if appropriate.

LONG DESCRIPTION

Displays a long description that can be blanks and can be duplicated, if appropriate.

24BIT

Displays the amount of 24-bit requested memory, except for one case where it is a count.

ANY

Displays the amount of any requested (usually 31-bit) memory, except for one case where it is a count.

64BIT

Displays the amount of any requested 64-bit memory. 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.

OCCURRENCES

Displays the number of occurrences of LUW, STME, STMR, and XCL (see the descriptions in the following table).

PROGRAM MEMORY (AT EOJ)

In z/OS, this line shows the amount of memory for most (but not all) programs.

DBMEMPR CALLS

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.

ERROR, NON-SQL

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 CA Support. Users have no control over it.
- ERROR, SQL is a parameter error on a memory manager request. This counter is used only by CA Support. Users have no control over it.

MAIN and MA31

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.

Remaining lines

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
CDC	Change Data Capture	Working storage for Change Data Capture
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.
IX64	INDEX BFRS -------	Index buffers in 64-bit memory.

DESC	LONG DESCRIPTION	Explanation
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.
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.
STMC	STMT PREPARE: CURSOR UPDSQL	SQL statement prepare - cursor update.
STMD	STMT EXECUTE: DYN-CURSOR	SQL statement execution - dynamic cursor.

DESC	LONG DESCRIPTION	Explanation
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 MUF 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.

```
                                    MUF EOJ LOG STATUS INFORMATION

MUF EOJ, LOG CLOSED, CLEAN EOJ
LOG CONTROL BYTES-229,376 ROLLBACK-3,963
LOG RECORD BYTES-22,373,107 WRITTEN-24,891,580
LOG RECORD SEQ-3 LOG BLOCK SEQ-243
```

This page of the report provides the following information:

MUF EOJ, LOG CLOSED, CLEAN EOJ

This message indicates that there was a *clean* EOJ. If the EOJ had not been clean, other messages would have appeared here.

LOG CONTROL BYTES

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.

LOG RECORD BYTES

This line gives LOG RECORD BYTES *n*, where *n* can be up to 31 digits, which is 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, 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.

LOG RECORD SEQ

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.

MUF EOJ Report Logging Section Changes

The MUF LXX format options feature is a performance enhancement that allows MUF to process more maintenance requests per second while reducing the total cost of ownership. For details, see the *CA Datacom/DB Database and System Administration Guide*.

In support of the LXX format options feature, the logging section of the MUF EOJ provides new headings and six new detail lines as follows:

- One detail information line was added for each of the following: ADDIT, DELET, and UPDAT.
- One detail information line was added for TSN records, that is, transaction-related information such as for the following (some of which are internally used and nowhere externally described):
  - COMIT
  - ROLBK
  - CLOSE
  - LOGCP
  - LOGCR

- DEQUE
- LOGPR
- LOGPN
- LOGRP
- LOGBA
- LOGBU
- LOGBD
- YYC??
- DELAL

- One detail information line was added for moved records in support of compressed tables or the DBUTLTY OLREORG function.
- One detail information line was added for OTHER records, including LOGIT, LOGDW, OPEN, and a CLOSE that was not done as a commit.

Note: Many of the fields in the report are provided to show that features are working and their results, largely for use by CA Support, CA Datacom development, and QA debugging.

All lines show the total record count as follows:

- Counts of 0-99999 are printed as specific numbers
- Counts of 100,000-9,999,999 are printed as rounded to 4 digits with the letter k
- Counts of less than 9 gigabytes (<9g) are rounded to 4 digits and printed with the letter m
- Numbers greater than 9 gigabytes (>9g) are rounded to 4 digits and printed with the letter g

Other lines in the report are as follows:

PRCT OF LXX

provides the byte count of the records as a percent of all log bytes during this MUF execution, with one decimal place if it amounts to one percent or higher. If the size is less than one percent, however, it is printed as a number similar to the total record count field previously described.

Note: Other lines in the report are formatted as appropriate to their descriptions, similarly to the PRCT OF LXX format.

OLD RCD EXP

provides the bytes of the log records that contained the before image record in fully expanded format. Includes encrypted tables written as encrypted.

OLD RCD CMP

provides the bytes where CA Datacom/DB compression was used.

OLD RCD USER

provides the bytes where user compression was used and the option allowed them to be written as compressed (includes 16 bytes of compression name and data).

NEW WA ORIG

provides the bytes during a DELET or UPDAT that were the original user Work Area that was logged as smaller than the other options.

NEW WA DELTA

provides the bytes during an UPDAT that were logged as a Work Area with only changes based upon the LOG_RECORD_UPDAT_SIZE option.

NEW RCD EXP

provides the bytes that were logged as the expanded after record. Includes encrypted tables written as encrypted.

NEW RCD CMP

provides the bytes that were logged as the compressed after record.

NEW RCD USER

provides the bytes that were logged as the user compressed after record.

ELEM LIST

provides the bytes that were logged as the element list (FORMAT 0 external style or FORMAT 1 internal style) for records logged with a Work Area.

FORMAT 1 SAVED

provides the bytes that were saved by records logged over how they would have been saved with FORMAT 0.

In the following example, the number of log bytes was 337 million, which makes each percent 3m. The numbers in the form *nn.n* are percentages, and the others are less than one percent, which means they are printed as a number of bytes.

```
            PRCT --   OLD   OLD   OLD   NEW   NEW   NEW   NEW   NEW      FORMAT
      TOTAL    OF --   RCD   RCD   RCD    WA    WA   RCD   RCD   RCD  ELEM     1
GROUP  RCDS   LXX --   EXP   CMP  USER  ORIG DELTA   EXP   CMP  USER  LIST SAVED
ADDIT  352K  30.7                     1262K         6.6 3289K  727K  165K   1.6
DELET  352K  32.1      8.0   1.2 1705K                                     1453K
UPDAT  352K  36.2      8.0   1.1  844K 1262K   2.2  1008   146     0   1.7   6.1
TSN    7781 1685K
MOVER  3340 1042K
OTHER   304 84805
```

Buffer Pool Information

```
                    BUFFER POOL INFORMATION
NAME    NUMBER    SIZE  USED 1  USED 2  USED 3  USED 4  USED 5+


IXX         12      4K       7       5       5       4     199-K
DXX        210      4K    2543    1707    1539    1453     269-K
DATA       210      8K       2       0       0       0         0
DATA2       20  32,767       0       0       0       0         0

IXX01       10      8K       0       0       0       0         0
DXX01       20      8K       0       0       0       0         0
DATA01      40     32K    6479    6182    6088    6005     293-K
IXX02       10      8K       0       0       0       0         0
IXX03        9      4K       0       0       0       0         0
DXX03        9      4K       0       0       0       0         0
DATA03       9      4K       0       0       0       0         0
DATA99      40      4K       0       0       0       0         0
```

This page of the report provides details about what buffer pools exist and their 1-5 usage. This report includes both the standard buffer pools and the alternate buffers pools, if defined to the current MUF.

Note the following:

DATA2

The DATA2 line only appears if the current MUF has a standard large data pool defined.

SIZE

The numbers in the buffer SIZE column are not rounded, that is, the numbers represent the actual buffer size.

USED

The buffer USED totals are rounded numbers that follow standard CA Datacom rules regarding when to round numbers to K, M, or G.

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.

```
                                CBS INFORMATION

                SETS PROCESSED              561       SETS DELETED MAX AGE         0
                SETS SPILLED                  0       SETS FETCHED                 0
                CBSBFR CURRENT PERCENT FULL   0       CBSBFR MAX PERCENT FULL      0
```

This page of the report shows an example of the additional CBS information in the EOJ report:

SETS PROCESSED

Displays the total number of sets processed by the MUF.

SETS DELETED MAX AGE

Displays the number of sets which were deleted from the CBS buffer due to the CBS MAX AGE setting.

SETS SPILLED

Displays 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 want to either increase the CBS buffer size or examine your applications for missing SELPR statements.

SETS FETCHED

Displays 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.

CBSBFR CURRENT PERCENT FULL

Displays the percent of CBS buffer max size that is being used by open sets.

CBSBFR MAX PERCENT FULL

Displays the high water mark for concurrent open sets.

CDC Report Example

```
                                    CHANGE DATA CAPTURE



DB03161I - CDCL MUFP1 CLOSED LASTCDC-20070827100407 LASTLXX-20070827100709
DB03162I - CDCL MUFP1 POLL-15 GAPWAIT-  MODE-REMOTE(DS)
DB03163I - CDCL MUFP1 CDCLXXIO-73,682 MNT-785,000 WRAP-0
```

This page of the MUF EOJ report shows an example of CHANGE DATA CAPTURE (CDC. For more information about the CDC feature, see the *CA Datacom/DB Database and System Administration Guide.*

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.

```
                                    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.

```
                                    TASK INFORMATION

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
```

This page of the report provides the following information:

B-RF

Displays the current buffer references.

LOGR

Displays the number of log records executed by the task since the last close or commit occurred.

COMM NEWRXX (Request New Recovery File)

The COMM NEWRXX option requests a new Recovery File (RXX) be started. In response, the MUF performs an IBM CLOSE/OPEN on the Recovery File, allowing operations to mount a **new** tape.

Note: When running with active recovery (LOGRCV=YES) in z/OS, it is possible to specify a DASD file for the Recovery File (RXX). However, we recommend that you do not do this. The COMM NEWRXX function of DBUTLTY is ignored in this mode. Also, a DASD file that becomes full causes a system x37 abend (such as B37) and the MUF is unusable until canceled and restarted.

The COMM NEWRXX function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

When using a console command, these steps are unnecessary.

When to Use

Use NEWRXX when using an active Recovery File (MUF 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 MUF 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, MUF 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 MUF. 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.

MUF dynamically deallocates the current RXX DD and reallocates a new one. CA Datacom/DB closes and reopens the Recovery File (RXX). Operations is responsible for mounting a new tape.

Dynamic Allocation of Recovery File

When allocating the new Recovery File dynamically, the following DD statement parameters must be used.

DSN

The same DSN as the original RXX DD statement, except that if this is a generation DATA group, then the next number is requested.

DISP

Is specified as NEW and the same dispositions as the original RXX DD statement.

UNIT

Is TAPE except when the MUF startup option RXX provides a different value. This value is supplied at MUF startup.

For more information, see the *CA Datacom/DB Database and System Administration Guide*.

UNIT COUNT

Is provided only if specified in the MUF startup option RXX. This value is supplied at MUF startup.

For more information, see the *CA Datacom/DB Database and System Administration Guide*.

VOLUME COUNT

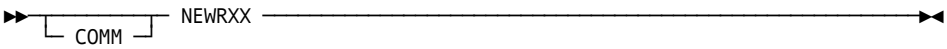
Is provided only if specified in the MUF startup option RXX. This value is supplied at MUF startup.

For more information, see the *CA Datacom/DB Database and System Administration Guide*.

How to Use the Console Command

The MUF must be active when you 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

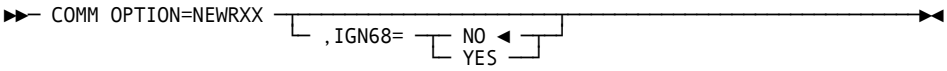
You can omit the COMM function name when submitting this console command.

NEWRXX

Requests the MUF to begin recording Recovery File information on a new tape.

How to Use the DBUTLTY Command

The MUF must be active when you execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keyword

OPTION=NEWRXX

Requests the MUF 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*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the 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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          COMM     OPTION=NEWRXX
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

FUNCTION=COMM
   OPTION=NEWRXX
```

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.

```
DB13105I - COMM     NEWRXX   REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF 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 MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

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

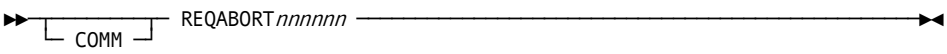
When you abort a request, CA Datacom/DB issues a return code 45 to the application program. Verify that programs contain instructions for handling the return code 45; for example, terminate further processing or have the program continue.

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 MUF Jobs) (see page 283). Use this request sequence number in the command to issue an abort request from the console.

The MUF must be active when you 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

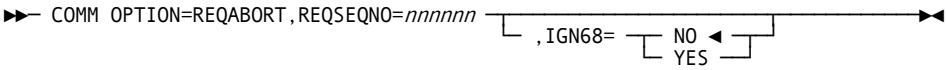
Sequence number of the request to be aborted.

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.

The MUF must be active when you execute this command.



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keywords

OPTION=REQABORT

Requests the MUF 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*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the 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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          COMM     OPTION=REQABORT,REQSEQNO=124019
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  COMM OPTION=REQABORT,REQSEQNO=00282

                  FUNCTION=COMM
                     OPTION=REQABORT
                     REQSEQNO=000282
```

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.

```
DB00604I - COMM     REQABORT REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF 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), or to SYSOUT if ML is specified in the SYSOUT MUF startup option. The dump contains the Master List information. You can run this option at any time.

Note: If you are running in a MUFplex environment, when COMM OPTION=SNAP is requested a CA Datacom/DB return code 94(104) is generated on the remote MUFs in addition to the SNAP dump on the local MUF. The return code is **94 (GENERAL ERROR INDICATOR)** with an internal return code of **104 ERROR OCCURRED ON ANOTHER MUF WITHIN THE MUFPLEX** at HEX offset 15A (master list dump reason). For details, see return code 94(104) in the *CA Datacom/DB Message Reference Guide*.

The COMM SNAP function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

When using a console command, these steps are unnecessary.

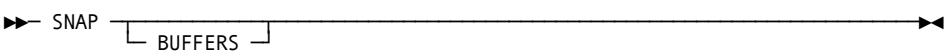
When to Use

Use this option at the direction of CA Support. If you are having a problem with the MUF, CA 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

The MUF must be active when you 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

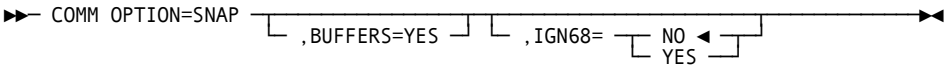
BUFFERS

Requests that the standard CA Datacom/DB buffer area be included.

Note: If 64 bit index buffers are used, using SNAP BUFFERS (or DBUTLTY COMM OPTION=SNAP,BUFFERS=YES) results in an SVC dump (SDUMP) instead of the normal PXX dump.

How to Use the DBUTLTY Command

The MUF must be active when you execute this command. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the MUF.

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.

Note: If 64 bit index buffers are used, using DBUTLTY COMM OPTION=SNAP,BUFFERS=YES (or the SNAP BUFFERS console command) results in an SVC dump (SDUMP) instead of the normal PXX dump.

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

Note: 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 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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          COMM     OPTION=SNAP,BUFFERS=YES
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
COMM OPTION=SNAP,BUFFERS=YES

FUNCTION=COMM
   OPTION=SNAP
   BUFFERS=YES
```

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.

```
DB13105I - COMM     SNAP REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF 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 SNAPCSA (Print Dump of Common Storage Area)

The COMM SNAPCSA option prints a dump of the CA Datacom/DB related area in your system Extended Common Storage Area (ECSA). The SNAPCSA option is for z/OS systems with a MUF environment. You can run this option at any time. It is not necessary that the MUF be active.

When to Use

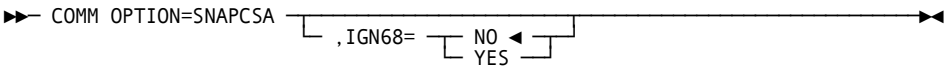
Use if CA Support requires the ECSA snap.

SNAPCSA Output

This print is directed to the SNAPER data set. Its associated ID is 002.

How to Use

You can execute this command in either Single User or with the MUF active. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keywords

OPTION=SNAPCSA

Requests that the utility produce a snap dump of the CA Datacom/DB Extended Common Storage Area (ECSA).

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

Note: 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 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 print a dump of the CA Datacom/DB related area in your system ECSA.

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.
//SNAPER   DD SYSOUT=*
 //SYSIN    DD *                                  Command input
          COMM    OPTION=SNAPCSA
 /*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
JOB DBDVLCSA         STEP CSA            TIME 121301   DATE 2001226  ID = 002    CPUID = FF1719663090   PAGE 00000001

  PSW AT ENTRY TO SNAP    078D1000  8002B716  ILC  02  INTC  0033

-STORAGE

     LINE 02B82460  SAME AS ABOVE
02B82480 00000000 00000000 00000000 C4C2C4E5    D3404040 0000D8A3 00010001 C534F0F2   *............DBDVL   ..Q.....E.02*
02B82500 2CC4C2F0 F0F1F0F2 C9406040 C5D5C4C5    C4404040 D1D6C260 C4C2C4E5 D3404040   *.DB00102I . ENDED   JOB.DBDVL   *
     LINES 02B82560-02B825C0  SAME AS ABOVE
                      .                                      .                                  .
                      .                                      .                                  .
                      .                                      .                                  .
```

This page of the report shows a portion of a dump produced with the SNAPCSA option.

COMM SNAPSVC (Print Dump of the SVC)

The COMM SNAPSVC option prints a dump of the SVC. CA Datacom Version 14.0 does not, however, expect an SVC to be installed. Therefore, this ability to dump should not be needed.

When to Use

Use this option only at the direction of CA Support.

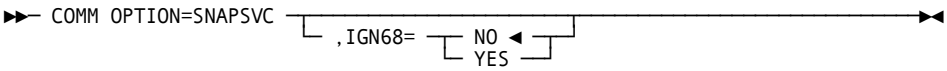
This option is valid only in a MUF environment. It is not necessary for the MUF to be active to execute this option.

SNAPSVC Output

The print is directed to the SNAPER data set and its associated ID is 001.

How to Use

You can execute this command in either Single User or with the MUF active. Execute DBUTLTY with the command:



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keyword

OPTION=SNAPSVC

Requests that the utility produce a snap dump of the SVC area.

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.

Note: 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 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 print an SVC dump to the SNAPER data set:

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
          COMM    OPTION=SNAPSVC
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Sample Report: COMM SNAPSVC

```
JOB DBDVLCSA         STEP CSA            TIME 121817   DATE 2001226  ID = 001    CPUID = FF0719663090   PAGE 00000001

  PSW AT ENTRY TO SNAP    078D1000  8002B6A6  ILC  02  INTC  0033

-STORAGE

34C69260                                                18B6186E 47F0B1C0   *                        ...>.0.¤*
34C69280 23400000 C4C2E2E5 C3D7D940 40404040    F1F24040 405C4040 4040C4D6 D9E2E5C3   *. ..DBSVCPR     12   *    DORSVC*
                      .                                   .                                .
                      .                                   .                                .
                      .                                   .                                .
```

This page of the report shows a portion of a dump produced with the SNAPSVC option.

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.

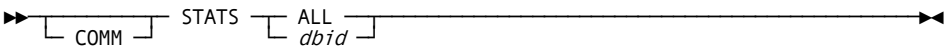
COMM STATS Flushes Pipeline

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

The MUF must be active when you 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.

STATS

Requests that current statistics be written.

Required Keywords

ALL

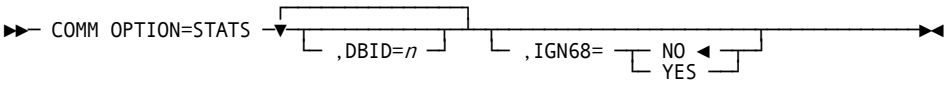
Specify ALL, if you want all databases to be processed. If you request that STATS be written for all databases, STATS are written for all databases that are *not virtual* databases. If any databases are virtual databases, a warning message is issued.

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

The MUF must be active when you execute this command. Execute the COMM function with the following command format.



Command

COMM

Invokes the function to pass communication to the MUF.

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*.

The DBSYSID macro parameter DELAY68= can be used to specify the number of minutes that the 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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
         COMM      OPTION=STATS
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

 FUNCTION=COMM
    OPTION=STATS
```

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.

```
DB13105I - COMM     STATS    REQUEST ACCEPTED
```

This page of the report shows a message indicating that DBUTLTY successfully placed the request on the MUF communication queue.

COMM STATUS (Print Status of MUF Jobs)

The COMM STATUS option displays the status of all jobs operating through the MUF. 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 MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

When using a console command, these steps are unnecessary.

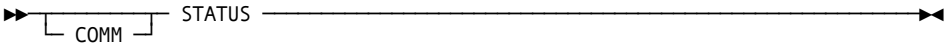
When to Use

Use the COMM STATUS option when you want to display the status of a job.

How to Use the Console Command

The MUF must be active when you execute this command. The console STATUS command provides substantial information but not all of the information that is provided using DBUTLTY COMM STATUS.

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

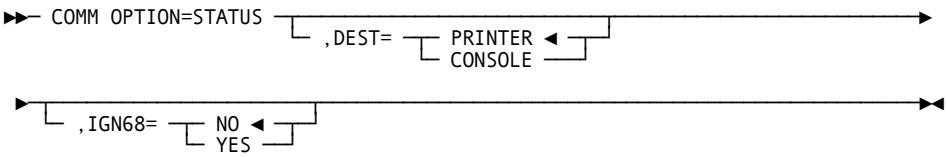
You can omit the COMM function name when submitting this console command.

STATUS

Requests a display of the current tasks in the MUF.

How to Use the DBUTLTY Command

Execute the COMM function with the following command format:



Command

COMM

Invokes the function to pass communication to the MUF.

Required Keyword

OPTION=STATUS

Requests that the current status of tasks in MUF be displayed.

Note: In z/OS, when DBUTLTY is executing on the same System as the MUF, the data is returned directly from memory with no MUF request. Also note that the DBUTLTY COMM OPTION=STATUS provides more information than the console STATUS, and this additional information may or may not be important to see.

Optional Keywords

,DEST=

Specifies the destination for the status report. CONSOLE causes the report to appear on both the console and SYSPRINT. PRINTER causes the report to appear on SYSPRINT 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.

Note: 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 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 print the status of all jobs operating through the MUF.

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.
//SYSIN    DD *                                  Command input
          COMM     OPTION=STATUS
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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

                FUNCTION=COMM
                   OPTION=STATUS
```

The 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.

```
MULTI-USER DBDVM02  AVAILABLE         TASKS - ATTACHED-----1, AVAILABLE----24

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
```

This page of the report shows the following:

MULTI-USER DBDVM02 AVAILABLE

This is a message indicating the MUF job status.

JOBNAME

Displays the name of the active job currently connected to the MUF 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

*****CDCL***

A Change Data Capture Listener (CDCL) task

*****CDCM***

A Change Data Capture Monitor (CDCM) task

*****DBRST**

RESTART function during the enable of the MUF

*****DBSYS**

Rare systems functions

*****HISTR**

History database spill

*****IOCMP**

Asynchronous I/O completion system processing

*****MPLXY**

MUFplex system processing - normal

*****MPLXZ**

MUFplex system processing - abnormal

*****QUEUE**

Index queue

*****SCRTY**

Security

*****SPILL**

Log spilling

*****STAR**

CA Datacom STAR system

*****STARX**

CA Datacom STAR user requests

R-UNIT

Displays the run unit number, where nnnnn is the low order five digits with leading zeros suppressed (nnnnn is zero for system tasks).

TASK

Displays the task number nnnn (RWTSA ID) where the task is running.

STATUS

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).

ANOTHER MUFPLEX

A task is waiting on another MUF in the MUFPLEX to take an action.

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.

CDC TSN RC14

A CDCU program is waiting for a TSN record to be added.

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.

HISTORY SPILL

Either a close is waiting on the history spill task, or the history spill task is waiting on a close.

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.

MUFPLEX LIST CF

A task is waiting on an outstanding XES list request.

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 MUF.

PLEX LIST DBAR

Directory (CXX) area control record, MUFplex Coupling Facility lock. With this condition, the DBID field contains the specific base required, and the table name field contains the specific area required.

PLEX LIST DBCR

Directory (CXX) base control record, MUFplex Coupling Facility lock. With this condition, the DBID field contains the specific base required.

PLEX LIST DBFC

Log Force Area (FXX), MUFplex Coupling Facility lock.

PLEX LIST DBLC

Log control/current blocks, MUFplex Coupling Facility lock.

PLEX LIST DBLF

Directory (CXX) table control record, MUFplex Coupling Facility lock. With this condition, the DBID field contains the specific base required and the table name field contains the specific table required.

PLEX LIST DBLG

Log add record, MUFplex Coupling Facility lock.

PLEX LIST DBUL

Log uncertainty block, MUFplex Coupling Facility lock.

PLEX LIST DBYJ

TSN, MUFplex Coupling Facility lock.

PLEX LIST DBYR

Run unit record, MUFplex Coupling Facility lock.

PLEX LIST DBYU

Assigned lock index, MUFplex Coupling Facility lock.

PLEX LOCK CF

A task is waiting on an outstanding XES lock request.

PLEX MSG REPLY

A task has sent a message to another MUF in the MUFplex and is waiting on a response from that MUF.

POSTED

The MUF 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 MUF request.

VARIABLE LOG

A task is waiting on access to the Log Area LXX control.

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 BFR LOCK C

A task is waiting to access a CXX block on another task owning lock.

WAIT BFR LOCK D

A task is waiting to access a data block on another task owning lock.

WAIT BFR LOCK E

A task is waiting to add an expand buffer on another task owning lock to add an expand buffer

WAIT BFR LOCK I

A task is waiting to access an index IXX block on another task owning lock.

WAIT BFR LOCK L

A task is waiting to access a Log Area LXX block on another task owning lock.

WAIT BFR LOCK X

A task is waiting to access an index DXX block on another task owning lock.

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 CDCL POLL

This Change Data Capture Listener task is waiting for work.

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 SP MG

A task is waiting on the Data Space Management lock.

WAIT DB MISC

A task is waiting to own DB miscellaneous lock on another task owning lock.

WAIT DBIO 24BIT

A task is waiting to add another DBIO 24-bit control block on another task owning lock.

WAIT DBLC LOCK

A task is waiting on the logging current block lock on another task owning lock

WAIT DBUL LOCK

A task is waiting on the log uncertainty list (DATACOM STAR) lock on another task owning lock

WAIT DBYJ LOCK

A task is waiting on the TSN Coupling Facility record on another task owning record.

WAIT DBYU LOCK

A task is waiting on the YU Coupling Facility record on another task owning record.

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 HASH STAR

A task is waiting on the HASH list within Datacom STAR on another task owning the list.

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 MRDF (C/V)

A task is waiting on the COVERED/VIRTUAL area lock on another task owning the lock.

WAIT MU-DSF

A task is waiting on a MUF DSF control lock on another task owning the lock.

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 PLEX ABEND

A MUF is in the process of being enabled in a MUFplex environment, and it is waiting on a prior MUF to finish recovering from an ABEND.

WAIT PLEX BDE

A task is waiting on MUFplex processing for a particular BDE.

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 REBUILD CF

A task is waiting on the Coupling Facility rebuild process on another task owning the lock.

WAIT REQ

A task is waiting on a request in another task area to complete before executing a spill.

WAIT RL-PLEX-JN

A task is waiting on DBID that is transitioning to shared in a MUFplex.

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 SOURCE

A task is waiting on an SQL resource on another task owning the resource.

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 STAR 2PHAS

A task is waiting on STAR two-phase commit lock on another task owning the lock.

WAIT STAR SCHED

A task is waiting on access to CA Datacom STAR scheduling.

WAIT STAR STOR

A task is waiting on access to CA Datacom STAR storage.

WAIT STAR

A task is waiting on CA Datacom STAR distribution.

WAIT SUBTASK

A task is waiting on a subtask completion.

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 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 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 XCF RQST

A task is waiting on an XCF request lock on another task owning the lock.

XCF RRS RESET

A task is waiting on an RRS RESETon another task owning the lock.

CMD

Displays the command in the Request Area. This might be blank for a system task.

DBID

Displays a database ID (not listed when not applicable).

TBL

Displays a table name (not listed when not applicable).

SEQ

Displays 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.

TIME

Displays the length of time in minutes and seconds since the MUF 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.

I/O

Displays 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.

ID

Displays (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 MUF, 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).

TSN

Only displayed on updates after a Transaction Sequence Number has been assigned.

OWNR/Y TSN-TM

Displays the task of the owner (OWNR) and the number (Y) of the MUF 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.

JOBNAME

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 MUF or, if a system task, to one of the entries listed in the table for the JOBNAME column on Page 2 of the report.

R-UNIT

Displays the run unit number, where *nnnnn* is the low order 5 digits with leading zeros suppressed (*nnnnn* is zero for system tasks).

TASK

Displays the task number *nnnn* (RWTSA ID) where the task is running.

VALUE/RID

Displays 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
CA DMS	CA-Disk has serialized a process within this MUF.	Variable (format unknown to CA Datacom/DB)
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 queue/backward read processing.	Database ID and Block group number
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

Type of Value Lock	Meaning	Additional Information Available from the COMM STATUS Display
DBSPLN	CA Datacom/DB has locked an SQL plan for concurrency.	AUTHID and Plan name
DBSPRC	The locking of a procedure 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
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.	

The following is also printed per task with the JOBNAME, R-UNIT, and TASK fields, followed by the new 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:

```
JOBNAME R-UNIT TASK BREF R-TIME C-TIME D-TIME W-TIME  PATH U-SYSTEM  U-JOBID USER-ID                    EOJ_OK
 DBDVLSTS 44431    1                  *NULL* *NULL* UNKN CA11     JOB39119 COMM                       YES
```

BREF

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.

R-TIME

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.

C-TIME

C-TIME is the CPU time that the current request (SQL, SAAT, or 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.

D-TIME

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.

W-TIME

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.

PATH

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).

U-SYSTEM

U-SYSTEM is the system name, where the user application called the interface.

U-JOBID

U-JOBID is the z/OS JOBID identification of the user application.

USER-ID

USER-ID is the 32-character user ID, if this request was made through the DBNTRY entry point.

EOJ_OK

EOJ_OK is YES or NO as specified (for details, see the DBURSTR parameter EOJ_OK= in the *CA Datacom/DB Database and System Administration Guide*).

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
```

COMM TASK_ABORT (Abort a Task)

The COMM TASK_ABORT option provides a means of terminating certain tasks open in the MUF.

The COMM TASK_ABORT function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the CA Datacom/DB Database and System Administration Guide for more information on XCF.

When using a console command, these steps are unnecessary.

When to Use

Use this option to terminate a single-task inactive connection to the MUF.

A typical use of this facility is to abort an inactive TSO task associated with a CA product that uses CA Datacom/AD (AD) as its data repository. Upi may have completed the use of the AD-based product, but left the connection open. Such open connections can prevent bringing the MUF down or closing an associated URT.

Note: TASK_ABORT has no effect on tasks with an active status, tasks with a wait status, or tasks from multi-task connections to the MUF such as those by CA Datacom CICS Services and CA Datacom Server. Additionally, It is not effective for tasks attached remotely using XCF or CCI.

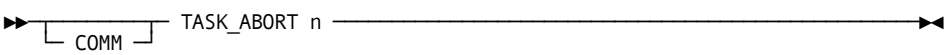
The task number that you specify in TASK_ABORT must be connected to the MUF when you execute this command. Ensure that you specify the desired task number in the TASK_ABORT command. If the task meets the criteria, TASK_ABORT will abort it. If the task is simply a long running task, you probably do not want to abort it.

How to Use

To use the console command, issue a STATUS command or Dynamic System Table query to MUF_ACTIVE_TASKS to obtain the task number of the task that you want to abort. For details, see COMM STATUS (Print Status of MUF Jobs) or the *System Tables Reference Guide.* Use this task number in the command to issue an abort request.

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: The MUF must be active when you execute this command.



Command

COMM

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

TASK_ABORT

Requests an abort of a task that was connected to the MUF as a single task and currently in 'NOT ACTIVE' status.

Required Keywords

n

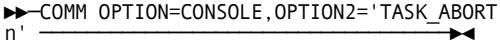
Task number of the task to abort.

How to Use the DBUTLTY Command

To execute the TASK_ABORT function using DBUTLTY:

1. Perform a STATUS command to determine the task number of the task you want to abort. Alternatively issue a Dynamic System Table query to MUF_ACTIVE_TASKS to obtain this information.
2. Issue the following command to terminate the request, where 'n' is the task number.

Note: The MUF must be active when you execute this command.



Chapter 10: CONFIRM (Verify Constraint Integrity)

Constraint integrity can be verified by using the CONFIRM function of the CA Datacom/DB Utility (DBUTLTY).

The MUF must be initiated to perform this function.

This function communicates with one MUF. Be aware of the following:

- Verify that DBUTLTY is communicating with the desired MUF by executing DBUTLTY with the same System Identifier module (DBSIDPR) that is used by the MUF.
- Verify proper load library concatenation by keeping the System Identifier modules (DBSIDPR) in separate load libraries.

If the z/OS Cross-System Coupling Facility (XCF) is being used, verify that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information about XCF.

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 519)), the LOAD portion of the REORG function, (for details, see REORG (Parallel BACKUP/LOAD) (see page 623)), and the RECOVERY function (for details, see RECOVERY (Rebuild a Database) (see page 591)).

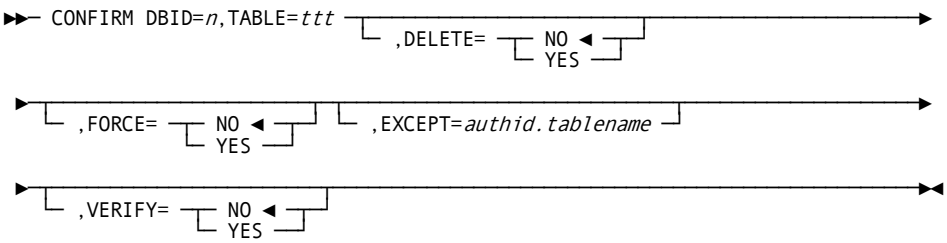
Note: You must CONFIRM tables in the following order, from referenced-to-referencing, or parent-to-child, that is to say, the referenced table cannot be in check pending status. For example, if there is a foreign key from a REMARKS table to a LINE_ITEM table on PO_Nbr and Line_Item_Nbr, and the LINE_ITEM 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

The MUF must be active when you 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

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.
//SYSIN    DD *                                  Command input
  CONFIRM DBID=30,TABLE=DAT,VERIFY=YES,DELETE=YES,EXCEPT=DB.OPEN
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CONFIRM DBID=30,TABLE=DAT,VERIFY=YES,DELETE=YES,EXCEPT=DB.OPEN

FUNCTION=CONFIRM
   DBID=30
   DELETE=YES
   EXCEPT=DB.OPEN
   TABLE=DAT
   VERIFY=YES
```

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.
- 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 11: CXXCLONE (Clone Environment)

The main purpose of the CXXCLONE function is to help you clone an environment to a new environment that is modeled after an existing environment.

When to Use

Use the CXXCLONE function when you want to clone an environment to a new environment that is modeled after an existing environment.

The CXXCLONE function, depending on the options specified, can restore one database, a range of databases, or a list with any combination of databases and ranges. Alternately, if no database ID is specified, every database of input is added to the current CXX.

Consider the following example. If you had an environment named TEST5, and you had a reason for needing another environment you decided to name TEST6, one step in the creation of the new environment would be to build a TEST6 CXX. In this case, using a LOAD AREA=CXX from a backup of the TEST5 CXX would leave every area in a loaded status and with the data set names from the TEST5 environment. The new TEST6 environment, however, needs its own data sets for every index and data area. Leaving all the areas in loaded status, and with the TEST5 data set names, risks that they might not get corrected and be used by both the TEST5 environment and the TEST6 environment, leading to a very large numbers of error conditions.

Using CXXCLONE to do CXX cloning avoids the problems just described because one strategy of using CXXCLONE is to set all the loaded tables to a status of not loaded and to do a deletion of all data set names. Using that capability, you can plan to create new data sets for the entire TEST6 environment during the initialize and load process. This is a safe approach, because the data for the loads can come from data backups of TEST5 data areas or elsewhere, such as the installation "clean" databases for system areas such as CA Datacom Datadictionary (usually databases 2 and 15).

Another general CXX cloning strategy is to copy all of the database data sets after building the new CXX. With this strategy, the data sets can be left loaded, but the data set names need to be altered from TEST5 to TEST6, or they should be deleted and the new names established using the DBUTLTY function CXXMAINT with OPTION=ALTER and DSN=. Also with this strategy, the data sets need to be updated to be "linked" to the new CXX named TEST6, and this is done with the DBUTLTY function LINK.

The DBUTLTY CXXCLONE function is similar to the LOAD AREA=CXX function of DBUTLTY in that it accepts a backup of a Directory (CXX) as an input data set and loads that data to an output CXX. But CXXCLONE and LOAD AREA=CXX are different in the following ways:

- CXXCLONE provides potentially useful options not available with LOAD. When you are not just restoring a backup of a CXX to the same CXX, CXXCLONE provides options you can use to make changes during the load.

  Note: If your goal is to make the CXX larger by doing a backup, we recommend that you use an INIT and a LOAD, using LOAD AREA=CXX.

- By using the STATUS= keyword of CXXCLONE, you can choose how you want the loaded status handled. But when using a LOAD, if the database being loaded does not exist, the tables are set as not loaded and, if it exists, the status is left unchanged (usually loaded).

- CXXCLONE allows input of a CXX of the same release or a CXX of the prior release. The LOAD function requires as input a backup of the same release as itself. CXXCLONE therefore supports a function similar to the DBUTLTY function CXXMAINT OPTION=CONVERT1214 but with the additional options detailed in the following syntax diagrams and descriptions.

Note: A CXX load done using CXXCLONE is treated by External Security the same as a LOAD AREA=CXX and therefore uses LOAD.CXX (full load, no table rights) or LOAD.CXXBASE (DBID(s) specified, CATALOG DTUTIL table rights).

Using CXXCLONE in the SIMPLIFY Mode

When using SIMPLIFY mode, when no DBID is specified the function is required to execute with the MUF not enabled. It requires that the CXX has just been initialized with the INIT function.

When at least one DBID is specified, the function expects to execute using the MUF, and all access to the CXX is controlled by and done by the MUF. However, the function might execute with the MUF not enabled and byopening and using the using the CXX in this address space, if the CXX is protected by either having this function follow a previous function that must run with MUF not enabled (such as INIT AREA=CXX), or having the MUF declared down using a function SET OPTION1=MUF_NOT_ENABLED. It is also possible to follow a function SET OPTION=MUF_ENABLED_OR_DISABLED that indicates the function should run with MUF if enabled or without MUF using a CXX locally.

Using CXXCLONE when not in the SIMPLIFY Mode

When not in the SIMPLIFY mode, CXXCLONE executes similarly to the LOAD function with AREA=CXX, in that it executes without the MUF using the CXX locally, if no DBID is specified, and with MUF and not any CXX locally, if a DBID is specified.

Note: In addition to the information just given about the CXXCLONE function and the SIMPLIFY mode, also see the information about CXXCLONE in the section on the SIMPLIFY feature in the *CA Datacom/DB Database and System Administration Guide*.

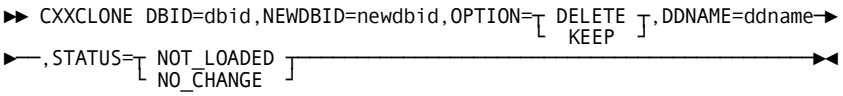
How to Use

There are four varieties of the CXXCLONE function of DBUTLTY.

CXXCLONE *case1* Syntax

Following is a syntax diagram for the CXXCLONE function of DBUTLTY that executes with:

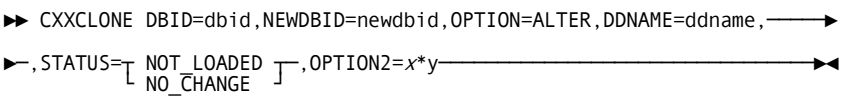
- a DBID
- a new DBID
- a data set name that is either kept or deleted



CXXCLONE *case2* Syntax

Following is a syntax diagram for the CXXCLONE function of DBUTLTY that executes with:

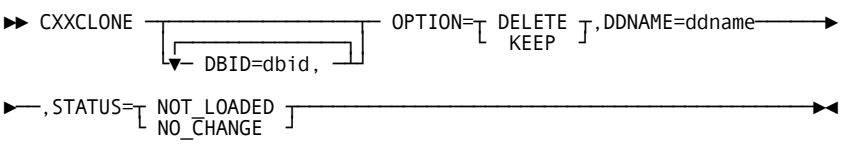
- a DBID
- a new DBID
- data set name alterations



CXXCLONE *case3* Syntax

Following is a syntax diagram for the CXXCLONE function of DBUTLTY that executes with:

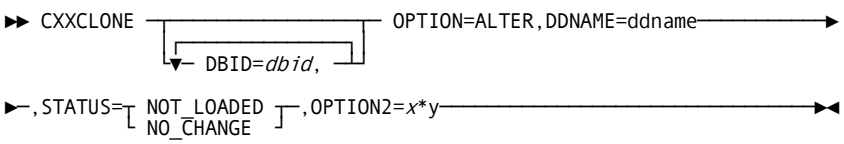
- an optional DBID
- no new DBID
- a data set name that is either kept or deleted



CXXCLONE *case4* Syntax

Following is a syntax diagram for the CXXCLONE function of DBUTLTY that executes with:

- an optional DBID
- no new DBID
- data set name alterations



Keyword Descriptions

DBID=*dbid*

(Required: case1 and case2) The *dbid* in DBID=*dbid* in *case1* and *case2* syntax is a single database ID, that is, you can only specify one DBID, not a range of DBIDs in *case1* and *case2* syntax.

(Optional: case3 and case4) If a DBID= is specified in *case3* and *case4* syntax, it can be specified as a single database ID or a range of database IDs, for example DBID=4-5. Also, if you specify DBID=, you can specify more than one occurrence, for example DBID=4,DBID=5,...DBID=*n*.

If DBID= is omitted in *case3* and *case4* syntax, every database of input is added to the current CXX.

DDNAME=*ddname*

(Required) The *ddname* in DDNAME=*ddname* is a one to eight (1-8) character DD name that points to a CXX backup for either Version 14.0 or Version 12.0.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. For information about how to disable the edit, see the note about DDNAME= check protection that follows.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

NEWDBID=*newdbid*

(Required: case1 and case2) The NEWDBID= keyword can only be used in *case1* and *case2* syntax. Use NEWDBID= to add (load) to the CXX being used by the current execution of DBUTLTY a single database with a database ID different from the one specified with DBID=. Therefore, the *newdbid* in NEWDBID=*newdbid* is a single database ID, that is, you can only specify one new database ID, not a range of database IDs.

OPTION=

(Required) The OPTION= keyword allows you to specify KEEP to keep the data set name from the input without changing It, DELETE to cause the data set name to be set to blanks (spaces), or ALTER to use the OPTION2= keyword (see following) to alter the data set name.

OPTION2=*xy**

(Required: case2 and case4) The OPTION2= keyword is required (and only allowed) when OPTION=ALTER (see previous) is specified. The OPTION2= value allows you to specify a data set name string (comprised of a set of contiguous bytes), where the string (represented by *x* in the syntax diagram) is followed by a single asterisk (*), followed by a replacement string, being a second set of a contiguous bytes (indicated by *y* in the syntax diagram). The combination of the keyword OPTION2= and its value must fit on a single input statement as is standard for DBUTLTY. This option is useful to those users following a best practice of having the CXX name within each data set within a database or the full CXX.

For example, a user cloning a CXX named MUF6 to be replaced by one named MUF007 would specify OPTION2=MUF6*MUF007. However, if a single string is not in every data set name present, then OPTION=ALTER cannot be used, in which case we recommend using OPTION=DELETE to remove the data set names and force each data set to be initialized, where the appropriate data set name can then be specified as part of that process. The *x* value can be longer or shorter or the same as the *y* value, but if the resulting data set name is longer than 44 bytes, the function fails immediately with error message DB13285E and leaves the database not processed at all or only partially processed. The function also fails immediately with error message DB13284E if the expected string is not present in each and every data set name provided by the input, including every index area and every data area with a known DSN (not blanks), for example where the CXX name is not used as part of data set names for all areas.

STATUS=

(Required) The STATUS= keyword is used to force the areas that are loaded to the CXX to either have the LOADED status of not loaded or allow the status to remain the same as the input. For that purpose, STATUS= can be specified as NOT_LOADED or NO_CHANGE.

Note: If STATUS=NOT_LOADED is specified, every table is set as not loaded, the index is set as not loaded, and the index areas and data areas are set as not initialized.

Examples

Example 1:

Build a test copy of production full CXX setting tables as not loaded and removing all data set names.

1. Using the production CXX DD statement:

   ```
   BACKUP AREA=CXX,DDNAME=x
   ```

2. Using the new test CXX DD statement:

   ```
   INIT AREA=CXX,DDNAME=x,DATACOM=x,CXXNAME=TEST
   CXXCLONE DDNAME=x,STATUS=NOT_LOADED,OPTION=DELETE
   ```

Example 2:

Build a full copy of production including CXX and all other data sets.

1. Using the Production CXX DD statement:

   ```
   BACKUP AREA=CXX,DDNAME=x
   ```

2. Using the new test CXX DD statement:

   ```
   INIT AREA=CXX,DDNAME=x,DATACOM=x,CXXNAME=TEST2
   CXXCLONE
   DDNAME=x,STATUS=NO_CHANGE,OPTION=ALTER,OPTION2=PROD*TEST2
   ```

3. Use IEBGENER to copy all data sets not the CXX to copies where the data set name in production of PROD is replaced by TEST2.

   The report provides a list of bases that were processed, such as:

   ```
   CXXCLONE DDNAME=X,DBID=497-697,DBID=797,STATUS=NO_CHANGE,OPTION=DELETE
   BASE PROCESSED - 497
   BASE PROCESSED - 597
   BASE PROCESSED - 697
   BASE PROCESSED - 797
   REQUEST COMPLETE
   ```

In addition to CXXCLONE, the BASE PROCESSED information has been added to the following functions:

- BACKUP AREA=CXX
- LOAD AREA=CXX
- CXXMAINT OPTION=CONVERT1214
- CXXMAINT OPTION=CONVERT1412

Note: The ability to use CXXCLONE or LOAD AREA=CXX with multiple occurrences of DBID= (including the use of ranges) are expected to be rare requirements, and the input file is processed from the start to the DBID= value for every candidate database. For example, whether the databases exist in the range or not, DBID=51-100 is processed by the input file 50 times.

If a NEWDBID= keyword is used in an execution of the CXXCLONE function, the additional information is also printed as shown in the following example report line:

```
BASE PROCESSED - 6     NEW DBID - 4,007
```

Example JCL

The following shows the use if the CXXCLONE 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. It is also possible that the CXX DD statement might need to be added for some combination of options.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=0M
//STEPLIB    See the previous note and JCL Requirements.
//Z        DD DSN=cxx.bkup,DISP=(,CATLG),         CXX backup data set
//SYSIN    DD *                                   Command Input
        CXXCLONE DDNAME=Z,STATUS=NO_CHANGE,OPTION=KEEP
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Sample Report 1

Following is a sample report when the command is:

```
CXXCLONE DDNAME=Z,STATUS=NO_CHANGE,OPTION=KEEP
```

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXCLONE DDNAME=Z,STATUS=NO_CHANGE,OPTION=KEEP

FUNCTION=CXXCLONE
   DDNAME=Z
   OPTION=KEEP
   STATUS=NO_CHANGE
```

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  DATACOM  VERSION
DBDVL      22,943  2010/12/03 07.57.30  DBDVM0           DB       14.0

                   BASE PROCESSED - 2
                   BASE PROCESSED - 6
                   BASE PROCESSED - 15
                   BASE PROCESSED - 16
                   BASE PROCESSED - 17
                   BASE PROCESSED - 97
                   BASE PROCESSED - 197
                   BASE PROCESSED - 297
                   BASE PROCESSED - 397
                   BASE PROCESSED - 497
                   BASE PROCESSED - 597
                   BASE PROCESSED - 697
                   BASE PROCESSED - 796
                   BASE PROCESSED - 797
                   BASE PROCESSED - 897
                   BASE PROCESSED - 997
                   BASE PROCESSED - 1,000
                   BASE PROCESSED - 1,006
                   BASE PROCESSED - 1,007
                   BASE PROCESSED - 2,009
                   *                                    REQUEST COMPLETE
```

Sample Report 2

Following is a sample report when the command is:
CXXCLONE DDNAME=Z,STATUS=NOT_LOADED,OPTION=DELETE

```
                   CONTROL CARD(S)
                   .........1.........2.........3.........4.........5.........6.........7.........8
                   CXXCLONE DDNAME=Z,STATUS=NOT_LOADED,OPTION=DELETE

                   FUNCTION=CXXCLONE
                      DDNAME=Z
                      OPTION=DELETE
                      STATUS=NOT_LOADED
```

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  DATACOM  VERSION
DBDVL      22,943  2010/12/03 07.57.30  DBDVM0           DB       14.0

                   BASE PROCESSED - 2
                   BASE PROCESSED - 6
                   BASE PROCESSED - 15
                   BASE PROCESSED - 16
                   BASE PROCESSED - 17
                   BASE PROCESSED - 97
                   BASE PROCESSED - 197
                   BASE PROCESSED - 297
                   BASE PROCESSED - 397
                   BASE PROCESSED - 497
                   BASE PROCESSED - 597
                   BASE PROCESSED - 697
                   BASE PROCESSED - 796
                   BASE PROCESSED - 797
                   BASE PROCESSED - 897
                   BASE PROCESSED - 997
                   BASE PROCESSED - 1,000
                   BASE PROCESSED - 1,006
                   BASE PROCESSED - 1,007
                   BASE PROCESSED - 2,009
                   *                                    REQUEST COMPLETE
```

Sample Report 3

Following is a sample report when the command is:

```
CXXCLONE DDNAME=Z,STATUS=NOT_LOADED,OPTION=ALTER,OPTION2=DBDVM0*DBDVM7
```

```
                   CONTROL CARD(S)
                   .........1.........2.........3.........4.........5.........6.........7.........8
                   CXXCLONE DDNAME=Z,STATUS=NOT_LOADED,OPTION=ALTER,
                   OPTION2=DBDVM0*DBDVM7

                   FUNCTION=CXXCLONE
                      DDNAME=Z
                      OPTION=ALTER
                      OPTION2=DBDVM0*DBDVM7
                      STATUS=NOT_LOADED
```

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  DATACOM  VERSION
DBDVL      22,943  2010/12/03 07.57.30  DBDVM0           DB       14.0

                   BASE PROCESSED - 2
                   BASE PROCESSED - 6
                   BASE PROCESSED - 15
                   BASE PROCESSED - 16
                   BASE PROCESSED - 17
                   BASE PROCESSED - 97
                   BASE PROCESSED - 197
                   BASE PROCESSED - 297
                   BASE PROCESSED - 397
                   BASE PROCESSED - 497
                   BASE PROCESSED - 597
                   BASE PROCESSED - 697
                   BASE PROCESSED - 796
                   BASE PROCESSED - 797
                   BASE PROCESSED - 897
                   BASE PROCESSED - 997
                   BASE PROCESSED - 1,000
                   BASE PROCESSED - 1,006
                   BASE PROCESSED - 1,007
                   BASE PROCESSED - 2,009
                   *                                    REQUEST COMPLETE
```

Sample Report 4

Following is a sample report when the command is:

CXXCLONE
DDNAME=Z,STATUS=NOT_LOADED,NEWDBID=998,DBID=997,OPTION=ALTER,OPTION2=DEV*QA

```
                   CONTROL CARD(S)
                   .........1.........2.........3.........4.........5.........6.........7.........8
                   CXXCLONE DDNAME=Z,STATUS=NOT_LOADED,NEWDBID=998,DBID=997,
                   OPTION=ALTER,OPTION2=DEV*QA

                   FUNCTION=CXXCLONE
                      DBID=00997
                      DDNAME=Z
                      NEWDBID=00998
                      OPTION=ALTER
                      OPTION2=DEV*QA
                      STATUS=NOT_LOADED
```

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  DATACOM  VERSION
DBDVL      22,943  2010/12/03 07.57.30  DBDVM0           DB       14.0

                   BASE PROCESSED - 997   NEW DBID - 998
                   *                                    REQUEST COMPLETE
```

Chapter 12: CXXMAINT (Maintain Directory)

Use the CXXMAINT function when you want to:

- Activate/deactivate a key for Compound Boolean Selection use
- Set Directory (CXX) locking to single image or Sysplex
- Alter the DSN specified in the Directory
- Activate/deactivate logging
- Convert the Directory to the current version
- Update Directory definitions from CA Datacom Datadictionary
- 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 Datadictionary 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 databases, CXXMAINT terminates without performing any modification with the following exceptions. You can use the following CXXMAINT ALTER options with a database open in a MUF:

- CBSUSE
- DSN
- DSOP
- LOGGING

You can also use CXXMAINT DELETE with a database open in a MUF as long as the specified table has no open User Requirements Tables. The ACCESS function can still be used, to prevent tasks from opening a database while you are running a CXXMAINT function. For more information, see ACCESS (Restricting Opens or Maintenance.

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 171) for details.

CXXMAINT ALTER CBSUSE (Trial Delete a Key from CBS Use)

The CXXMAINT 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.

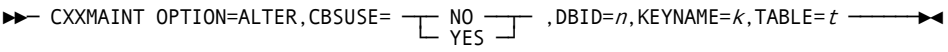
You can use this option with a database open in a MUF.

Effects on Maintenance

Adds or deletes of Index entries during maintenance continue during the trial deletion of a key.

How to Use

The MUF must be active when you 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.

```
//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
         CXXMAINT   OPTION=ALTER,CBSUSE=NO,DBID=1,KEYNAME=POLI,TABLE=POL
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=ALTER,CBSUSE=NO,DBID=1,KEYNAME=POLI,TABLE=POL

FUNCTION=CXXMAINT
   CBSUSE=NO
   DBID=1
   KEYNAME=POLI
   OPTION=ALTER
   TABLE=POL
```

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.

```
*                                  REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT ALTER CXXENQ (Controlling ENQ)

When running with the Simplify option, an executing MUF protects the CXX during the enable process and maintains it for the entire execution. This ability to provide one protection process is part of the performance benefits of the option. If the MUF is not enabled, DBUTLTY executions use Simplify protections and also the normal not-Simplify protections described in the information that follows.

When not running with the Simplify option or when the MUF is not enabled, locking the Directory (CXX) is done during all CXX control functions. The CXXMAINT ALTER CXXENQ option is part of the process of locking the CXX.

When to Use

Use this option to control the mode of how the Directory is locked. LOCAL mode does DASD locking for all Systems with shared DASD to the CXX and also locks the Directory with z/OS ENQ only on the MVS image where the code is executing. This option **is b**est if all CA Datacom execution is not within one Sysplex. It does have a risk of occasionally returning code 67, requiring a reexecution of the process that got the return code 67. SYSPLEX mode does no DASD locking and also locks the Directory with z/OS ENQ on all MVS images in the Sysplex where the code is executing. This option is best if all CA Datacom execution is within one Sysplex. It provides no protection of execution outside the Sysplex. With the recommended Simplify mode in use, the option should be set as SYSPLEX.

Regardless of choice, the CXX name is strongly recommended (required) to be unique within the entire connected site z/OS Systems. When using CXXENQ=LOCAL, we strongly recommend that each CXX name in the local z/OS environment be unique and, when using CXXENQ=SYSPLEX, we strongly recommend that each CXX name in the entire Sysplex be unique.

The setting for CXXENQ is provided on the full CXX report when no DBID= option is provided, or on the CXX report with TYPE=K. An example is: CXX ENQ – SYSPLEX.

How to Use

The MUF must be active when you execute this command. Execute DBUTLTY with the command:

Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,CXXENQ=

Specifies the Directory (CXX) locking mode.

CXXENQ=LOCAL specifies that the Directory is locked only on the MVS image where the code is executing.

Note: We strongly recommend that each CXX name in the local z/OS environment be unique.

CXXENQ=SYSPLEX specifies that the locking of the Directory is done on all MVS images in the Sysplex where the code is executing.

Note: We strongly recommend that each CXX name in the entire Sysplex be unique.

Sysplex locking has more overhead than local locking. If the MUF jobs, Single User jobs, and DBUTLTY jobs that open the Directory locally execute on the same MVS image, Sysplex locking is not necessary.

If the Directory is used with local locking on more than one MVS image, it is possible that return code 67 (CXX INTERLOCK) could be generated, due to the DASD protection done after the ENQ has timed out (because of slowness in the image with the code executing that owns the lock). Sysplex locking is preferable, however, to random occurrences of return code 67. Therefore, we recommend CXXENQ=SYSPLEX for sites executing for this Directory on more than one MVS image.

With this option, the Sysplex ENQ is exclusively used to protect the CXX, and the DASD locking is disabled. This prevents any possible occurrence of a return code 67. It requires that the MUF and all DBUTLTY functions execute in the same Sysplex. This option should be considered as being required if running as part of a MUFplex to ensure full protection from a return code 67.

Valid Entries:

LOCAL or SYSPLEX

Default Value:

(No default)

CXXMAINT ALTER DBCS (Activating DBCS Support)

The CXXMAINT 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 340).

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 Datadictionary 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

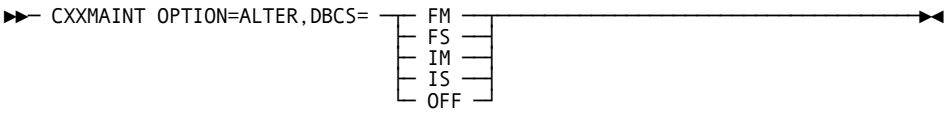
Use the ALTER DBCS option to enable and specify the type of DBCS support or to disable the DBCS support.

How to Use

The MUF must be active when you 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.

```
//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
         CXXMAINT   OPTION=ALTER,DBCS=IM
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    CXXMAINT OPTION=ALTER,DBCS=IM

                    FUNCTION=CXXMAINT
                       DBCS=IM
```

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.

```
                *                                   REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT ALTER DSN (Modify Data Set Name)

The CXXMAINT ALTER DSN option allows you to modify the DSN for reporting purposes.

When to Use

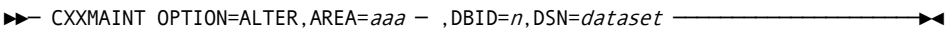
During the INIT function, CA Datacom/DB saves the data set name (DSN) for Index or data areas in the Directory. Use the CXXMAINT ALTER DSN option to modify the DSN for reporting purposes and, optionally, to change the DSN used for dynamic allocation. This eliminates having to reinitialize the Index or data areas to change the DSN in the Directory.

You can use this option with a database open in a MUF.

How to Use

The MUF must be active when you execute this command, except when running with the Simplify option and the function follows another that only runs with the MUF not enabled, or if you declare the MUF is down using a function: SET OPTION1=MUF_NOT_ENABLED or SET OPTION1=MUF_ENABLED_OR_DISABLED.

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 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.

If you want to remove a stored data set name, specify the data set name as an asterisk (DSN=*). Specifying an asterisk for the data set name sets the data set name to blanks and therefore removes the stored data set name.

Note: See restrictions on changing data set names below.

Valid Entries:

either a valid data set name or an asterisk (*)

Default Value:

(No default)

Note:

- The data set name for the base used for the CBS index can be changed only in one of the following ways:
  - The enabled MUF has a different database as the CBS index.
  - The MUF is not enabled, is not externally secured for DBUTLTY, and the DBUTLTY INIT function is used, or in Simplify mode it can be executed with MUF not enabled (see previous comments), 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
```

- 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 MUF has a virtual TTM base.
  - The enabled MUF has SQL not generated.
  - The MUF is not enabled, is not externally secured for DBUTLTY, and the DBUTLTY INIT function is used.
- 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.

```
//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
       CXXMAINT  OPTION=ALTER,AREA=DEM,DBID=1,DSN=DASUP.ALPHA.DEM001
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=ALTER,AREA=DEM,DBID=1,DSN=DASUP.ALPHA.DEM001

FUNCTION=CXXMAINT
   AREA=DEM
   DBID=1
   DSN=DASUP.ALPHA.DEM001
   OPTION=ALTER
```

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.

```
*                                  REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT ALTER DSOP (Modify Space Reclamation Option)

The CXXMAINT ALTER DSOP option allows you to change the space reclamation option specified for an area.

When to Use

Use the CXXMAINT ALTER DSOP 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 MUF.

Index Area Organization

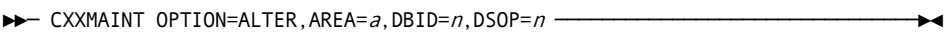
The ALTER DSOP option does not update the Index Area (IXX). The old option is still in effect for existing entries.

Synchronization with CA Datacom Datadictionary

You must also change the space management option attribute (SPACE-MNGMNT) for the area in CA Datacom Datadictionary. Otherwise, the old option is restored if the CA Datacom Datadictionary definition is subsequently cataloged to the CA Datacom/DB Directory.

How to Use

The MUF must be active when you 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 without deleted block index

2

Wraparound without deleted block index

3

Clustering

4

Basic space reclamation with deleted block index

5

Wraparound with deleted block index

Valid Entries:

0, 1, 2, 3, 4, or 5

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.

```
//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
        CXXMAINT   OPTION=ALTER,AREA=PAY,DBID=1,DSOP=0
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.......1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=ALTER,AREA=PAY,DBID=001,DSOP=0

FUNCTION=CXXMAINT
   AREA=PAY
   DBID=1
   DSOP=0
   OPTION=ALTER
```

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.

```
*                                REQUEST COMPLETE
```

This page of the report show a message indicating the maintenance completed successfully.

CXXMAINT ALTER LANGUAGE (Activating LANGUAGE Support)

The CXXMAINT ALTER LANGUAGE option allows you to specify whether to use the Katakana language support. Also see CXXMAINT ALTER DBCS (Activating DBCS Support) (see page 331).

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 Datadictionary 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.

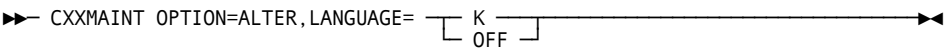
When to Use

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.

```
//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
         CXXMAINT   OPTION=ALTER,LANGUAGE=K
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    CXXMAINT OPTION=ALTER,LANGUAGE=K

                    FUNCTION=CXXMAINT
                       LANGUAGE=K
```

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.

```
                *                                   REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT ALTER LINK (Disable CXX Name Edit)

The CXXMAINT 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 important note in the following topic.

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.

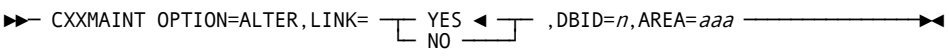
Some sites have a very large unchanging *reference* type database and want to have the same physical copy of it accessed by all of the various MUF systems. We do not recommend this. It can only be done safely if you follow these rules:

- 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 MUF. *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

The MUF must be active when you 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 note in CXXMAINT ALTER LINK (Disable CXX Name Edit) (see page 342).

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.

```
//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
        CXXMAINT   OPTION=ALTER,DBID=1,LINK=NO,AREA=PAY
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    CXXMAINT OPTION=ALTER,DBID=1,LINK=NO,AREA=PAY

                    FUNCTION=CXXMAINT
                       AREA=PAY
                       DBID=1
                       LINK=NO
                       OPTION=ALTER
```

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.

```
                    *                                   REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT ALTER LOGGING (Start/Stop Logging by Table)

The CXXMAINT ALTER LOGGING option allows you to activate or deactivate maintenance logging.

When to Use

Use ALTER LOGGING to activate or deactivate maintenance logging for any table that is defined in CA Datacom Datadictionary 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 MUF. Or, you can use the ACCESS function to prevent opens of the database containing a table that you want to change, with the instruction to wait until current tasks complete. For more information, see ACCESS (Restricting Opens or Maintenance).

Impact on Recovery

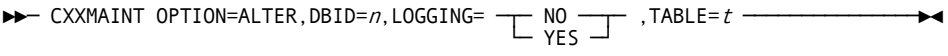
Carefully consider the consequences of turning logging on or off. Logging directly relates to CA Datacom/DB Recovery features, including transaction backout. See the *CA Datacom/DB Database and System Administration Guide* for details about logging.

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 Datadictionary definition and the DBUTLTY CXXMAINT option.

How to Use

The MUF must be active when you 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.

```
//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
        CXXMAINT   OPTION=ALTER,DBID=1,LOGGING=NO,TABLE=POL
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    CXXMAINT OPTION=ALTER,DBID=001,LOGGING=NO,TABLE=POL

                    FUNCTION=CXXMAINT
                       DBID=1
                       LOGGING=NO
                       OPTION=ALTER
                       TABLE=POL
```

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.

```
                    *                                   REQUEST COMPLETE
```

This page of the report shows a 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)).
- OPTION2=*areaev* specifies the data area data set extent validation option (see CXXMAINT ALTER OPTION2=areaev (Data Area Extent Validation)).
- OPTION2=*sumode* specifies that you allow Single User using the OPTION2=Y_SU or to disallow Single User using OPTION2=N_SU (see CXXMAINT ALTER OPTION2=sumode (Single User Mode)).
- OPTION2=MATCHSID specifies that you want to change the setting of Simplify mode, for example, if to allow easy testing in QA systems, you want to change the CXX SIMPLIFY option to match a current DBSIDPR module (see CXXMAINT ALTER OPTION2=MATCHSID (see page 355)).

CXXMAINT ALTER OPTION2=datafs (DATAFS Support)

A feature that reduces compression overhead 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. It is a best practice to have this option set to Y.

When to Use

Use the ALTER DATAFS option when you want to enable the DATA FAST SEARCH feature.

Previously, 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 search 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.

Reporting DATAFS in Full CXX Report

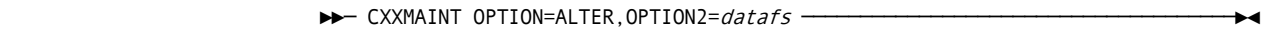
A full CXX report includes information on DATA FAST SEARCH *status*, where *status* in the report is either YES or NO.

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 algorithm 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)

This section discusses OPTION2=Y_DATAHU/N_DATAHU. It is a best practice to have this option set to Y.

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.

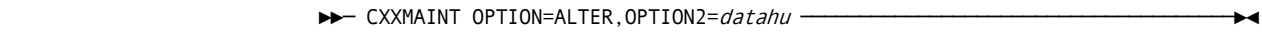
The runtime code in the MUF that maintains the next available overflow notices each time the next available overflow number crosses a 64-track boundary. If the memory copy of the high used mark is non-zero, MUF reads the data set control block. It then updates the high used mark and immediately writes 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 MUFs. 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)

This section discusses OPTION2=Y_AREAEV/N_AREAEV. It is a best practice to have this option set to Y.

When to Use

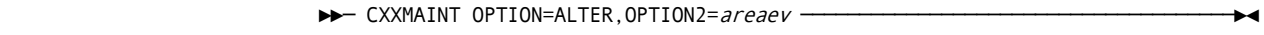
Use this option when you want to turn on or turn off data area data set extent validation.

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, use of Single User mode is blocked, even if secured Single User is allowed. We strongly recommend not using Single User mode.

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:

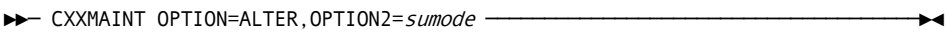
```
SECURE USING JOBNAME - NO                CXX LEVEL - 1
SECURE ALLOWING SINGLE USER - NO         SINGLE USER ALLOWED - NO
```

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. Previously, 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 about 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 OPTION2=MATCHSID

This section discusses CXXMAINT ALTER OPTION2=MATCHSID

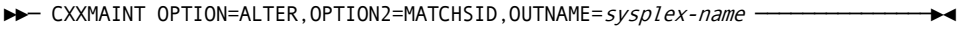
When to Use

To allow easy testing in QA systems, use this option when you want to change the CXX SIMPLIFY option to match a current DBSIDPR module. To use a CXX Data set on a different system in the Sysplex other than the one it is being initialized on, the Sysplex name in the CXX can be changed by including the OUTNAME= keyword.

How to Use

To change the CXX SIMPLIFY option to match a current DBSIDPR module, use the following command (the MUF must not be enabled when you run this function).

To change the CXX Sysplex name, use the following command (the MUF must not be enabled when you run this function).



Command

CXXMAINT

Invokes the function for CXX maintenance.

Required Keywords

OPTION=ALTER

Requests the CXXMAINT ALTER option.

,OPTION2=MATCHSID

Specify MATCHSID for OPTION2= when you want to change the CXX SIMPLIFY option to match a current DBSIDPR module to allow easy testing in QA systems.

The MUF must not be enabled when you run this function.

To be effective, this function needs to be the first in a DBUTLTY step that causes the CXX to open.

The CXXMAINT OPTION=ALTER,OPTION2=MATCHSID function causes a second DB00905I message. The first message shows the information from the CXX prior to change. The second message shows the information after the change. For more information about messages and return codes, see the *CA Datacom/DB Message Reference Guide*.

For more information about the SIMPLIFY feature, see the *CA Datacom/DB Database and System Administration Guide*.

Valid Entries:

MATCHSID

Default Value:

(No default)

Optional Keywords

,OUTNAME=*sysplex-name*

When a CXX is used on a Sysplex other than the one it was initialized on, OUTNAME= identifies the name of the Sysplex where the CXX will be used. The 'D XCF,SYSPLEX ALL' z/OS console command displays the Sysplex names of all the systems in a Sysplex.

Valid Entries:

A 1- to 8-characer name

Default Value:

(No default)

CXXMAINT ALTER SINGLE (Run in a Single User Environment)

The CXXMAINT ALTER SINGLE option allows you to run in a Single User environment with external security. At installation, the system is set to run in a MUF environment.

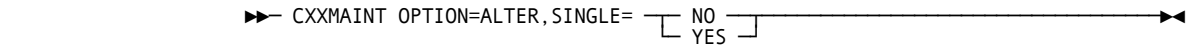
When to Use

Use ALTER SINGLE to execute Single User jobs with external security or, after executing this option, to run in a MUF environment. If you use this option, Single User jobs run *unsecured* and all MUF jobs run secured.

Note: 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 MUF 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.

```
//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
        CXXMAINT   OPTION=ALTER,SINGLE=YES
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=ALTER,SINGLE=YES

FUNCTION=CXXMAINT
   OPTION=ALTER
   SINGLE=YES
```

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.

```
*                                  REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT CONVERT1214 (Upgrade CXX to Current Version)

The CXXMAINT CONVERT1214 option of DBUTLTY upgrades your CA Datacom/DB Directory (CXX) from the previous version to make it compatible with the installed new version. The CONVERT1214 option reads the Directory backup from a prior version, alters it, and writes the Directory in the new format. For more information about upgrading, see the *CA Datacom Installation and Maintenance Guide*.

When to Use

Use CONVERT1214 only when upgrading the Directory from one version to another. This option does not allow you to skip a version. You must convert to a new version from the version immediately preceding.

The MUF must not be active and no other DBUTLTY is allowed to be executing when you execute this command. This is enforced with Simplify mode with several edits. Simplify mode prevents execution if the CXX has been opened by a previous function without the protection.

Backup Requirements

Since this option only reads data in the format produced by a backup of the previous release (Version 12.0, previously known as r12), you must have a backup of your CXX to run the CONVERT1214 option. Create the backup using the BACKUP function of the old version.

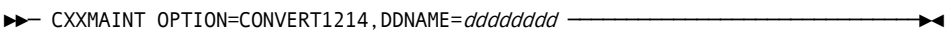
The backup that you use as input data for the CONVERT1214 option must be current, since all data in the Directory is replaced by data converted from the backup. The new Directory can be a different data set from the one from which the backup was made.

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

Execute DBUTLTY using the command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=CONVERT1214

Requests the CXXMAINT CONVERT1214 option.

,DDNAME=

Specifies the JCL DDname for the input data set containing the current Directory backup.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. We recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Input data set name

Default Value:

(No default)

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.

```
//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 (output)
//CXXIN    DD DSN=cxx.bkup,DISP=OLD              Previous version CXX backup
//SYSIN    DD *                                  Command Input
         CXXMAINT   OPTION=CONVERT1214,DDNAME=CXXIN
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=CONVERT1214,DDNAME=CXXIN

FUNCTION=CXXMAINT
   DDNAME=CXXIN
   OPTION=CONVERT1214
```

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.

```
*                                REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT CONVERT1412 (Downgrade CXX to Prior Version)

The CXXMAINT CONVERT1412 option of DBUTLTY downgrades your CA Datacom/DB Directory (CXX) from the current version to the prior version. The CONVERT1412 option reads the Directory backup, alters it, and writes the Directory in the old format.

This is the only Version 14.0 DBUTLTY execution allowed while pointing to a CXX set to Version 12.0.

When to Use

Use CONVERT1412 only when downgrading (falling back) the Directory from the current version to the prior version.

The MUF is not allowed to be active and no other DBUTLTY is allowed to be executing when you execute this command. This is enforced with Simplify mode with several edits. Simplify mode prevents execution if the CXX has been opened by a previous function without the protection.

Backup Requirements

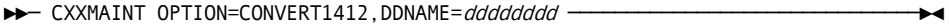
The backup must be taken with the current release.

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

Execute DBUTLTY using the following command:



Command

CXXMAINT

Invokes the function for Directory maintenance.

Required Keywords

OPTION=CONVERT1412

Requests the CXXMAINT CONVERT1412 option.

,DDNAME=

Specifies the JCL DDname for the input data set containing the current Directory backup.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. We recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Input data set name

Default Value:

(No default)

Example JCL

The following shows the command to downgrade 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.

```
//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 (output)
//CXXIN    DD DSN=cxx.bkup,DISP=OLD              Previous version CXX backup
//SYSIN    DD *                                  Command Input
         CXXMAINT   OPTION=CONVERT1412,DDNAME=CXXIN
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=CONVERT1412,DDNAME=CXXIN

FUNCTION=CXXMAINT
   DDNAME=CXXIN
   OPTION=CONVERT1412
```

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.

```
*                                REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT DDPROD (Update Directory)

The CXXMAINT DDPROD option updates the Directory (CXX) with CA Datacom Datadictionary definitions. This option reads the input file as created by CA Datacom Datadictionary and updates the Directory to reflect any changes, additions, or deletions.

CA Datacom Datadictionary also provides a means of cataloging CA Datacom Datadictionary definitions directly to the CA Datacom/DB Directory.

Note: For details about 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 Datadictionary 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 Datadictionary.
2. Execute the CA Datacom Datadictionary 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 Datadictionary 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 Datadictionary CATALOG transaction.

Active Processing

You can change an existing database definition as long as no task is actively processing the database. Use the ACCESS function (for details see ACCESS (Restricting Opens or Maintenance)) 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 Datadictionary definitions.

If any tables in the database being updated are not loaded, they are deleted from the Directory at the start of the process. The data set name is lost and to restore it, you must initialize the area or use the CXXMAINT OPTION=ALTER,DSN= utility function. 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 Datadictionary 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 Datadictionary 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 681).**

01

New table name

02

Area name change

08

Table ID change

09

Recover yes/no change

10

COMPRESSION CHANGED

11

RECORD LENGTH CHANGED

13

COMPRESSION NAME CHANGED

14

SPECIAL FIELD CHANGE

15

TO/FROM PARTITION

16

PARTITION COLUMNS UNEQUAL

17

PARTITION HIGH VALUES CHANGED

18

ANOTHER PARTITION CHANGED

19

ENCRYPTION CHANGED

- 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.

  Note: 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.

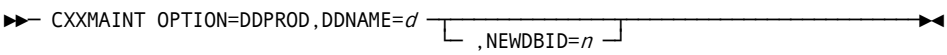
  If you are moving a table from one area to another, back up the area or extract the table before cataloging or using DDPROD to update the definition.

  Do **not** use the CXXMAINT DDPROD option for SQL accessible tables or on a database containing SQL accessible tables. Instead, use the CA Datacom Datadictionary 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

The MUF must be active when you 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 Datadictionary definitions.

,DDNAME=

Specifies the JCL DDname for the input data set.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

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 Datadictionary 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 Datadictionary 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.

```
//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
//DDCFBLD  DD DSN=dd.output,DISP=OLD          Input from CA Datacom Datadictionary
//SYSIN    DD *                                   Command Input
         CXXMAINT   OPTION=DDPROD,DDNAME=DDCFBLD
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=DDPROD,DDNAME=DDCFBLD

FUNCTION=CXXMAINT
   DDNAME=DDCFBLD
   OPTION=DDPROD
```

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.

```
                                IMPACT REPORT

BASE        AREA        TABLE ID        DISPOSITION
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
```

This page of the report shows a DBUTLTY impact report that gives 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 in Processing a DDPROD Request 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)

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 delete, those tables are deleted.

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, but it is subject to VIRTUAL_DELETE.

Deleting Tables

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 helps ensure that the Directory and the CA Datacom Datadictionary definitions are in synchronization and that the unwanted table definitions no longer exist in either CA Datacom Datadictionary or CA Datacom/DB.

1. Modify the entity-occurrence definitions in CA Datacom Datadictionary for the database structure, deleting the unwanted tables.
2. Execute the CA Datacom Datadictionary CATALOG function on the database.

SQL Accessible Tables

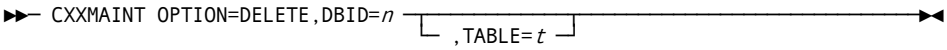
To delete tables accessed by SQL, use the SQL DROP TABLE statement which helps ensure that the Data Definition Directory (DDD) database remains synchronized with the Directory and CA Datacom Datadictionary. For more information, see the *CA Datacom/DB SQL User Guide.*

Note: A table can be deleted with the database open in the MUF, 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

The MUF must be active when you 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.

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.

```
//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
        CXXMAINT   OPTION=DELETE,DBID=238,TABLE=A38
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=DELETE,DBID=238,TABLE=A38

FUNCTION=CXXMAINT
   DBID=238
   OPTION=DELETE
   TABLE=A38
```

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.

```
*                                  REQUEST COMPLETE
```

This page of the report shows a 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.

```
//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
       CXXMAINT   OPTION=DELETE,DBID=238
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    CXXMAINT OPTION=DELETE,DBID=238

                    FUNCTION=CXXMAINT
                       DBID=238
                       OPTION=DELETE
```

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.

```
                *                                   REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT FORMAT0102 (Base Level Full Features)

Use the CXXMAINT FORMAT0102 function to specify the ID of the database or databases to alter from FORMAT 1 to FORMAT 2.

The FORMAT 1 ability provides features supported by Version 11.0 and later releases. The FORMAT 2 ability provides features supported by Version 12.0 and later releases. They were built to ease and speed upgrade from Version 11.0 to Version 12.0. All current Version 12.0 users should be at FORMAT 2 or convert to FORMAT 2. With Version 14.0, the two formats are supported but only expected to potentially be used by a site still running Version 11.0 and planning to upgrade to Version 12.0 only as a temporary pass through to Version 14.0. Version 14.0 is the last to support FORMAT 1.

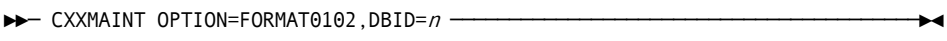
When to Use

When upgrading a Directory (CXX) from Version 11.0 to Version 12.0 to Version 14.0, you are expected to run in r11 compatibility mode for a few weeks and use only the Version 12.0 and Version 14.0 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 Version 11.0, Version 12.0, and Version 14.0. FORMAT 2 includes features supported by Version 12.0 and Version 14.0 but not by Version 11.0. When the time for running in Version 11.0 compatibility mode passes, all databases need to be set to FORMAT 2 to allow the additional Version 12.0 and Version 14.0 features not supported by Version 11.0. Those additional features include the following, in which Version 12.0 and Version 14.0.

- 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-Dataset Index area.

How to Use

The MUF must be active when you 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 databases 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.

```
//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
        CXXMAINT   OPTION=FORMAT0102,DBID=238
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=FORMAT0102,DBID=238

FUNCTION=CXXMAINT
   DBID=238
   OPTION=FORMAT0102
```

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.

```
*                                    REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

CXXMAINT FORMAT0201 (Base Level Restrict to r11 Compatible Features)

Use this function to specify the ID of the database or databases to alter from FORMAT 2 to FORMAT 1.

The FORMAT 1 ability provides features supported by Version 11.0 and later releases. The FORMAT 2 ability provides features supported by Version 12.0 and later releases. They were built to ease and speed upgrade from Version 11.0 to Version 12.0. All current Version 12.0 users should be at FORMAT 2 or convert to FORMAT 2. With Version 14.0, the two formats are support but only expected to potentially be used by a site still running Version 11.0 and planning to upgrade to Version 12.0 only as a temporary pass through to Version 14.0. Version 14.0 is the last to support FORMAT 1.

When to Use

User production and test systems are intended to have all databases in compatible FORMAT 1 during upgrading from Version 11.0 to Version 12.0 to Version 14.0 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 and Version 14.0 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 to allow for testing or use by QA.

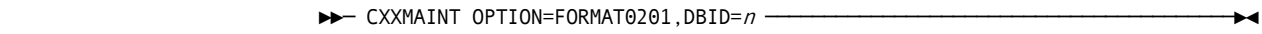
Considerations for Using FORMAT0201

Only a database in FORMAT 2 that has not used features of Version 12.0 and Version 14.0 has no issues with the change from FORMAT 2 to FORMAT 1. Issues exist for each database using Version 12.0-and Version 14.0 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 and Version 14.0 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

The MUF must be active when you 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 databases 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.

```
//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
        CXXMAINT   OPTION=FORMAT0201,DBID=238
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
CXXMAINT OPTION=FORMAT0201,DBID=238

FUNCTION=CXXMAINT
   DBID=238
   OPTION=FORMAT0201
```

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.

```
*                                  REQUEST COMPLETE
```

This page of the report shows a 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

Important! Do not use PURGE unless DELETE fails.

Before using PURGE, review CXXMAINT DELETE (Delete CXX Definitions) 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 Datadictionary.
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.

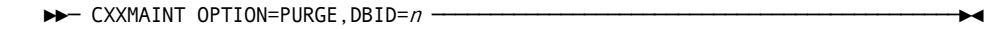
The PURGE option allows you to eliminate the problem database and restore only that database from a backup or rebuild it through CA Datacom Datadictionary.

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

The MUF must be active when you 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.

```
//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
        CXXMAINT   OPTION=PURGE,DBID=238
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    CXXMAINT OPTION=PURGE,DBID=238

                    FUNCTION=CXXMAINT
                       DBID=238
                       OPTION=PURGE
```

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.

```
                *                                   REQUEST COMPLETE
```

This page of the report shows a message indicating the maintenance completed successfully.

Chapter 13: 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 as follows:

Note: Although this program executes under the control of DBUTLTY, it does not execute under the special authority of DBUTLTY.

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=

Macro	Unsupported Parameters
DBURTBL	AREANAM= DBFLSUB=

Note: 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 next, 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.

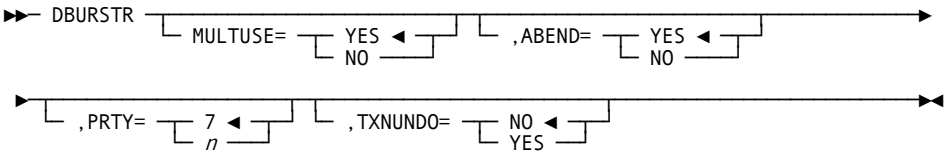
This function communicates with one MUF. You can use it with more than one MUF when you use the DBURSTR command and specify MULTUSE=YES. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

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.

YES specifies that CA Datacom/DB intercepts all abnormal terminations allowed by the operating system and performs the necessary termination processing.

NO specifies that CA Datacom/DB does not intercept abnormal terminations.

Valid Entries:

NO or YES

Default Value:

YES

MULTUSE=

Specifies whether the program directs requests to the MUF.

YES causes the MUF interface program to be loaded and all requests to be made through the MUF.

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 MUF.

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:

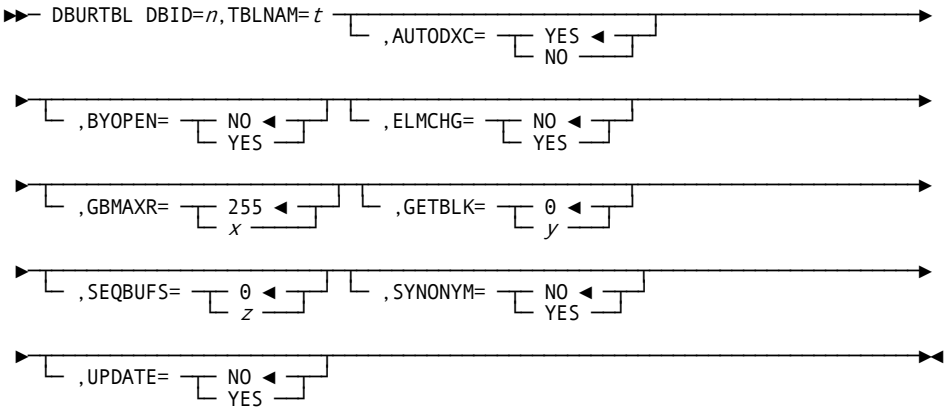
NO or YES

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 MUF 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=

(MUF 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:

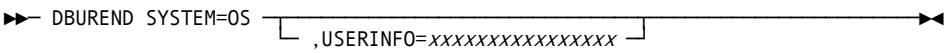
NO or YES

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/OS, specify OS.

Optional Keywords

,USRINFO=

Specifies one to sixteen characters of user information to be placed as a marker in this User Requirements Table.

Valid Entries:

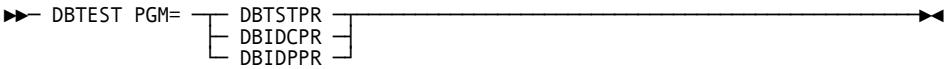
A 1- to 16-character string

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 393))
DBIDCPR (see description on DBTEST DBIDCPR (Index Analysis Program) (see page 398))
DBIDPPR (see description on DBTEST DBIDPPR (Index Population Analysis Program) (see page 400))

Default Value:

(No default)

DBTEST DBTSTPR (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 MUF does *not* recognize DBTSTPR as a DBUTLTY controlled job with special privileges.

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 SYSPRINT. **R** Print request area and work area in snap 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 CA 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 cumulative 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.
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 later in this table. The use of a number (1, 2, or 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.

```
//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
         DBURSTR  MULTUSE=YES
         DBURTBL  DBID=1,TBLNAM=PMF
         DBUREND  SYSTEM=OS
         DBTEST   PGM=DBTSTPR
DUMP=P
*CMD TBLKEYNM  DBI KEY           ELEM*4
LOCKGPMFSTZIP      TX75000
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
DBURSTR  MULTUSE=YES

FUNCTION=DBURSTR
   MULTUSE=YES
```

This page and the next three pages of the report show the following:

- The commands 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.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBURTBL  DBID=1,TBLNAM=PMF

                    FUNCTION=DBURTBL
                       DBID=1
                       TBLNAM=PMF
```

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBUREND SYSTEM=OS

                    FUNCTION=DBUREND
                       SYSTEM=OS
```

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBTEST   PGM=DBTSTPR

                    FUNCTION=DBTEST
                       PGM=DBTSTPR
```

```
DUMP=P                                                                          5/25/2003       12.01.50
*CMD TBLKEYNM  DBI KEY           ELEM*4                                                        12.01.50
LOCKGPMFSTZIP      TX75000                                                                     12.01.50
TX75243                                                                      LOCKGPMFSTZIP  12.01.50
```

This page of the report shows the following:

DUMP=P

Displays the dump option statement entered requesting input statement display.

ELEM

Displays one line of returned elements.

LOCKG

Displays the first command - a Locate Key.

TX75243

Key value returned from successful execution of the LOCATE KEY.

LOCKGPMFSTZIP

The request area followed by a return code.

12.01.50

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 CA 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 386).)
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 388).)
3. Code a DBUREND macro specifying the operating system in the SYSTEM= parameter. (See DBTEST Command 3 (DBUREND) (see page 391).)
4. Code the DBTEST function specifying the program (PGM=) as DBIDCPR. (See DBTEST Command 4 (DBTEST) (see page 392).)

Example DBIDCPR 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.

```
//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
         DBURSTR  MULTUSE=NO
         DBURTBL  DBID=1,TBLNAM=PMF
         DBUREND  SYSTEM=OS
         DBTEST   PGM=DBIDCPR
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBURSTR  MULTUSE=NO

                    FUNCTION=DBURSTR
                       MULTUSE=NO
```

This page and the next three pages of the report show 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.

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) and terminates.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
DBURTBL  DBID=1,TBLNAM=PMF

FUNCTION=DBURTBL
   DBID=1
   TBLNAM=PMF
```

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
DBUREND  SYSTEM=OS

FUNCTION=DBUREND
   SYSTEM=OS
```

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
DBTEST   PGM=DBIDCPR

FUNCTION=DBTEST
   PGM=DBIDCPR
```

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 CA 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 386).)
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 388).)
3. Code a DBUREND macro specifying the operating system in the SYSTEM= parameter. (See DBTEST Command 3 (DBUREND) (see page 391).)
4. Code the DBTEST function specifying the program (PGM=) as DBIDPPR. (See DBTEST Command 4 (DBTEST) (see page 392).)

Example DBIDPPR 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.

```
//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
         DBURSTR  MULTUSE=NO
         DBURTBL  DBID=1,TBLNAM=PMF
         DBUREND  SYSTEM=OS
         DBTEST   PGM=DBIDPPR
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBURSTR  MULTUSE=NO

                    FUNCTION=DBURSTR
                       MULTUSE=NO
```

This page and the next three pages of the report show 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.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBURTBL  DBID=1,TBLNAM=PMF

                    FUNCTION=DBURTBL
                       DBID=1
                       TBLNAM=PMF
```

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBUREND  SYSTEM=OS

                    FUNCTION=DBUREND
                       SYSTEM=OS
```

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    DBTEST   PGM=DBIDPPR

                    FUNCTION=DBTEST
                       PGM=DBIDPPR
```

```
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
    .          .           .    .      .
    .          .           .    .      .
    .          .           .    .      .


    .          .           .    .      .
    .          .           .    .      .
    .          .           .    .      .
    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
```

This page of the report shows the following:

LEVEL

Displays the Index level.

TOTAL PTR'S

Displays the total number of record pointers.

DATA CHANGES

Displays the number of key value data changes.

KID

Displays the Key ID.

LENGTH

Displays the length of the stored portion of the key data.

HIGH VALUE

Displays the highest key value in this block. (Only character type data is displayed.)

The second section

Displays the totals for the preceding block.

The third and final section

Displays the totals for the preceding block and grand totals.

Chapter 14: DEFAULT (Control Defaults)

By making use of the ability of the DBIN1PR Macro to process user configuration options prior to normal SYSIN processing, the DBUTLTY DEFAULT function allows you to control a few defaults.

Important! We recommend that you use the DBUTLTY DEFAULT function with caution. We recommend caution because when most users note a condition code of 0 (zero), they do not additionally view output. Therefore, other than expected results are not noticed. For that reason, it is simpler and of less potential confusion to all DBUTLTY users if the actual JCL streams are adjusted for site standards, instead of making use of the DEFAULT function.

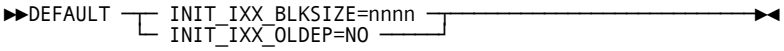
Note: DBIN1PR is a special process and does not follow the normal statement rules allowed by SYSIN. This includes placement of blanks and lack of the ability to have a continued statement.

When to Use

Use the DEFAULT function when you want to control the allowed defaults.

How to Use

Use an alternate (additional) SYSIN with a DBUTLTY DEFAULT function to control certain defaults. Execute the DEFAULT function using the following command format:



Command

DEFAULT

Invokes the function to control certain defaults.

Use no blank before DEFAULT when you specify it in columns 1-7. Follow DEFAULT with one blank (and use no continue lines).

Required Keywords

INIT_IXX_BLKSIZE=*nnnn*

INIT_IXX_BLKSIZE=*nnnn* gives you the ability to specify a user default *nnnn* for the block size during an INIT function with AREA=IXX used to initialize an index area (IXX or multiple data set index).

INIT_IXX_OLDEP=NO

INIT_IXX_OLDEP=NO gives you the ability to specify a user default for the Old Entry Point support for an index during an INIT function with AREA=IXX used to initialize an index area (IXX or multiple data set index). Only INIT_IXX_OLDEP=NO can be specified. The DBUTLTY default of YES is not valid for the DEFAULT function.

Example

In the case of an INIT_IXX_BLKSIZE=*nnnn*, the INIT of an index area (INIT AREA=IXX) allows an optional keyword of BLKSIZE= that can be specified as numbers from 4096 through 8192. The DBUTLTY default is 4096 or less, optimized for the device type containing the data set. If you want to replace the default with your own default of, for example, 5555, you could use an additional SYSIN with the following statement:

```
DEFAULT INIT_IXX_BLKSIZE=5555
```

For every DBUTLTY execution using this additional SYSIN, if an INIT is done with an AREA=IXX that does not have BLKSIZE= specified, it is given the user default of BLKSIZE=5555.

Consider the following:

- The DEFAULT function allows you to override this default. Therefore, if you provided a statement INIT AREA=IXX,BLKSIZE=8182, for that one INIT, the index would be initialized with a block size of 8192.
- An individual function has no path to requesting the normal default.
- The additional SYSIN can be overridden for any step by having a JOB/STEPLIB pointing to a different DBIN1PR module.
- If the DBIN1PR module indicates a data set, the additional SYSIN can be overridden for any step by having a DD statement pointing to a different input.
- All DBIN1PR input is echoed to the console without an option, for example: DB10096I - DBIN1PR PROCESSED - DEFAULT INIT_IXX_BLKSIZE=5555.
- The use of a default does not force an error.

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    INIT AREA=IXX,DBID=997

                   FUNCTION=INIT
                      AREA=IXX
*****- NOTE  -*****   BLKSIZE=05555  DB10028I - DBIN1PR DEFAULT APPLIED
                      DBID=00997
```

This page of the report shows that, to make the DEFAULT more visible in the report, a note and message are displayed to indicate that a DEFAULT has been applied.

Chapter 15: 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) and then doing a RETIX or LOAD of the DBID.

For the INIT function to operate on the IXX, the DBID must be completely offline to other users. However, the DEFRAG function runs through the MUF and allows the DBID to remain online and usable to other applications during the DEFRAG process.

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 fill. 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.

Locking

DEFRAG SMP locking is used to interact with backwards processing in the index, such as when a LOCBR command is issued. In such cases, DEFRAG obtains an exclusive SMP lock at the start of each key ID processing. If there are no backward requests for this key ID during the life of the DEFRAG, the lock is simply maintained. If there is a backward request during the process, DEFRAG notes where it is in the DXX and frees the SMP lock to allow the backward process to run. DEFRAG then reestablishes the lock, rereads its blocks, and continues as before. Backwards processing obtains a shared SMP lock. Information about DEFRAG SMP locking can be found in the SMP Informational Report obtained by specifying it with the SMPTASK MUF startup option. See the description of the DEFRAG event shown in the SMP Informational Report section.

Processing

The DEFRAG function is necessarily an I/O intensive 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 MUF 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 MUF 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).
- Cannot be run in a MUFplex environment if two or more MUFs are enabled.

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.

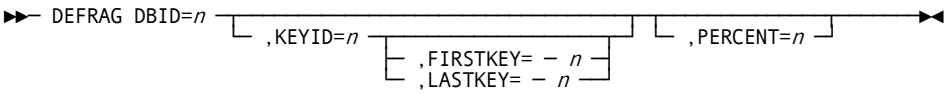
DEFRAG

When to Use

Use the DEFRAG function when you want to defragment the index (IXX).

How to Use

The MUF must be active when you 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=

Specify the DATACOM-ID of the key that you want to use for this execution of the DEFRAG function.

When you do not specify a KEYID=, all of the key IDs in the database are subjected to defragmentation.

If by analyzing past executions you know that certain key IDs do not benefit from defragmentation, you can exclude those key IDs from being defragmented by using KEYID= to specify a single key ID that does benefit from a DEFRAG execution.

If more than one key ID benefits from defragmentation, specify a separate DEFRAG execution for each of those key IDs.

Valid Entries:

3-digit DATACOM-ID of the desired key

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
         DEFRAG  DBID=781
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                         DEFRAG  DBID=781

                    FUNCTION=DEFRAG
                       DBID=00781
```

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.

```
                                        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 16: 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.

When to Use

Use the EDIT 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 with the MUF active. 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.

```
//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
         EDIT
         DBURSTR  ABEND=YES,MULTUSE=YES
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    EDIT

                    FUNCTION=EDIT


DB10016I - EDIT ONLY, NO FUNCTION(S) PROCESSED
```

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.
- Message DB10016I indicates an EDIT was in process.
- 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.

```
                    CONTROL CARD(S)
                    ........1.........2.........3.........4.........5.........6.........7.........8
                    DBURSTR ABEND=YES,MULTUSE=YES

                    FUNCTION=DBURSTR
                       ABEND=YES
                       MULTUSE=YES


DB10016I - EDIT ONLY, NO FUNCTION(S) PROCESSED
```

Chapter 17: ENCRYPT (Facilitate Data Encryption)

The main purpose of the DBUTLTY ENCRYPT function is to facilitate the data encryption feature of CA Datacom/DB. The data encryption feature allows tables to store encrypted data. This table-level option is available in z/OS environments only. The method of doing encryption and decryption is called basic encryption. For details about using data encryption, see the *CA Datacom/DB Database and System Administration Guide.*

With encryption, every DBUTLTY function that accesses rows of a table from (or to) DASD in the DBUTLTY address space must be done with an authorized execution of DBUTLTY. The same is true for setting or changing any encryption function.

Running not authorized generates an error code 'N' during READRXX, DBUTLTY SPLIT, RECOVERY, or REPORT AREA=RXX. The LXX flag indicating encrypted data can only be reset by an INIT or RESET of the LXX.

Important: The error code 'N' just mentioned is for any read of an RXX, no matter whether a particular RXX contains encrypted data. Termination occurs before access to any RXX data. CA Datacom does not bring potentially encrypted data into memory for an unauthorized user to attack.

Note: For DBUTLTY External Security, ENCRYPT has a function of ENCRYPT with no sub function and no table rights.

When to Use

Use the ENCRYPT function when you want to facilitate the data encryption feature of CA Datacom/DB by using the SET_BASIC_KEY_1 option.

SET_BASIC_KEY_1 sets the basic key 1 used for tables that are defined in that in CXX as using basic encryption.

Opening and Backing Up Encrypted Tables

Opening a Table that is Defined as Encrypted

During the opening of a table defined as encrypted, CA Datacom/DB performs an edit (validation check).

The edit continues with validation that the table definition being opened matches the definition either during the LOAD of the area or during the REPLACE of the table. This edit set is not for every facet of definition but for those facets that would clearly block the successful decryption of data. The current list of facets that must match (and that have not been changed) include the following:

- table name
- table ID
- row length
- use of DB compression
- use of user compression and its user data
- variable-length table (VSAM-T)
- recovery option
- encryption type and method

DBUTLTY LOAD and REORG BASIC Encryption Functions

Each DBUTLTY LOAD (or REORG) copies the current CXX key 1 information to the data area control block and uses that information for all basic encryption. A REPLACE function uses the basic encryption information in the control block, if it has been set. If not set, REPLACE copies the key 1 information from the CXX and sets the area as not openable by releases before Version 14.0.

In addition to the CXX encryption information, the data area has table level information also preserved to help ensure that data can be accessed correctly.

The addition of the CXX information and table information can cause the data block size to be insufficient to load the data. If this occurs, a return code is issued and the load terminated. This is likely only for areas with a high number of tables (the maximum is 240) and areas that have a small block size, for example, less than 4096.

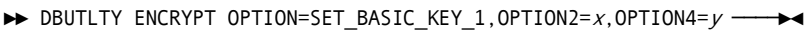
A data area loaded with encryption is set at a ‘level’ that is not openable by releases of CA Datacom/DB before Version 14.0, ensuring that back-level code does not try to read or add data out of sync with encryption options.

Similar to the REPORT AREA=CXX,TYPE=K that can print the encryption information stored in the CXX, the copy stored in the data area can be reported. Because they are intended to always match, this should be a rare need. As such, it is added to the REPORT TYPE=U function where a DD statement is provided to a data set considered as unknown. The execution for a data set that contains encrypted data formats the information similarly to the TYPE=K report.

How to Use

SET_BASIC_KEY_n Option

To set the basic key 1 that is used for tables defined, in that CXX, as using basic encryption, use the DBUTLTY ENCRYPT function as shown in the following syntax:



OPTION=SET_BASIC_KEY_1

(Required) Basic encryption key 1 is stored in the CXX and available to functions LOAD, REORG, and REPLACE as needed.

Value 1 is used with every DBUTLTY LOAD or REORG (with the LOAD option)), or REPLACE if the area has no other encrypted tables, done to an area that has an encrypted table and is the value stored in the data area control block to ensure successful access.

,OPTION2=***x***,OPTION4=***y***

(Required) We recommend that you request CA Datacom to generate a random key value. Request a random key value by specifying an asterisk (*) for both OPTION2= and OPTION4= (that is, OPTION2=*,OPTION4=*).

If you want to select the encryption key value of 32 bytes in length, specified as 64 hexadecimal digits (0-9 and A-F), use the options without the asterisks. The *x* is replaced by 32 digits representing the first half of the selected key. The *y* is replaced by 32 digits representing the second half of the selected key. If the *x* and *y* values are both set as all zeros, the values are never used as an encryption keys but are instead considered to be the absence of a key, which is therefore a way to delete a key.

The keys are considered "handles" of the actual encryption key to be used. The conversion of the handle or external key to the internal key is never intended to be made public. The key handles are saved in the CXX. When not setting the key to zeros to remove it, the values for OPTION2= and OPTION4= cannot have more than two repeating values and must have at least one occurrence of each possible hex value (0-9 and A-F).

A basic encryption key can only be set using an enabled MUF.

Examples

```
//DBUTLTY    EXEC PGM=DBUTLTY
*
 ENCRYPT OPTION=SET_BASIC_KEY_1,
 OPTION2=*,
 OPTION4=*
 ENCRYPT OPTION=SET_BASIC_KEY_1,
 OPTION2=1722EB3B75817A5A0787E6FB6B4D6C99,
 OPTION4=24A3841C6B365D0E3E17791F77110211
```

Example JCL

The following shows the command to use the basic ENCRYPT function.

Note: Use the following as a guide to prepare your JCL. The JCL statements are only examples. 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.
 //SYSIN    DD *                                  Command Input
         ENCRYPT OPTION=SET_BASIC_KEY_1,OPTION2=*,OPTION4=*
 /*
```

Sample Report

DBUTLTY provides a report that includes encryption information, including any basic encryption key information and the tables encrypted. The new report is requested by the REPORT function with a combination of keywords having AREA=CXX and TYPE=K. The report is available to all users and starts by providing the full CXX information also provided by a REPORT AREA=CXX without a DBID keyword and with either no TYPE= keyword or TYPE=B. After that information is provided, the known encryption information is provided. That encryption information is not considered a secret. Having this special report allows you to externally secure the use of this DBUTLTY function.

Note: For DBUTLTY External Security, REPORT AREA=CXX,TYPE=K uses REPORT.ENCRYPT with no table rights.

The DBUTLTY BACKUP AREA=CXX function backs up the encryption information stored in the CXX. The information restored to the CXX by either the LOAD AREA=CXX function or the CXXCLONE function when the options DBID is not specified. However, only if no DBID is provided to restore the entire CXX is the encryption key information restored to a CXX using DBUTLTY functions LOAD AREA=CXX and CXXCLONE.

Examples

Following is a sample report. For an example of the report header, see Sample Report Headers.

```
REPORT AREA=CXX,TYPE=K


CONTROL AREA DIRECTORY
DBCRBAS REQUIREMENT (ALL BASES OPEN)  -    53,360          CXX ENQ – LOCAL
NUMBER OF DATA BASES -    24                             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 - NO                         SINGLE USER ALLOWED – YES
SQL MODE - SUPPORT ACTIVE - NO                           SIMPLIFY MODE – YES
           DATADICTIONARY BASE -     0
           DATA DEFINITION DIRECTORY BASE -     0


ENCRYPTION HARDWARE SUPPORT - A(AES-128)=Y B(AES-192)=Y C(AES-256)=Y
ENCRYPTION INFORMATION - BASIC
KEY   - NOT USED
KEY 2 - NOT USED
DBID    TABLE  TYPE              METHOD


0789  C01    B(BASIC)           A(AES128)
0789  C02    B(BASIC)           B(AES192)
```

Following is another example.

```
ENCRYPTION INFORMATION - BASIC
KEY 1 - 041537850CEFFEDB6044E27F4A723139576293FF08E2772B4BFDAC9B7B4F1B16    ccyy/mm/dd
```

In the examples just shown, the following fields are defined as follows:

ENCRYPTION HARDWARE SUPPORT

The ENCRYPTION HARDWARE SUPPORT line in the TYPE=K report tells whether the hardware running the report supports the A, B, or C options. Attempts to use encryption without this support cannot be successful.

ENCRYPTION INFORMATION – BASIC

The ENCRYPTION INFORMATION – BASIC section in the TYPE=K report tells the current KEY 1 value. It is set using DBUTLTY function ENCRYPT with OPTION=SET_BASIC_KEY_1. If the key has not been set, it is reported as NOT USED, but if set the date last set is reported

Note: This is not the actual key being used but the external form of the key.

Table Information

The following section provides information about every table in the CXX that has encryption defined. But because the section only reflects encrypted tables, if none exist the section does not exist. When it does exist, however, it provides the DBID and table name with encryption. It also provides the type of encryption and method of encryption.

Note: Tables that are in history status and those not history status are listed (for a description of history status, see the full CXX report example).

Encryption Information in Full CXX Report without TYPE=A

The table information relating to encryption is provided as part of a full REPORT AREA=CXX without TYPE=A. An example of the changed lines are:

```
TABLE NAME - C01


.
RECOVER  - YES   ENCRYPTION - NONE     LOGGING  - YES
PIPELINE - YES


TABLE NAME - C02
.
RECOVER  - YES   ENCRYPTION - B(BASIC)           A(AES128)LOGGING  -
YES
PIPELINE - YES


TABLE NAME - F01
.
RECOVER  - YES   ENCRYPTION - B(BASIC)           B(AES192)LOGGING  -
YES
PIPELINE - YES
```

Chapter 18: 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 sets.

Important! Prior to the execution of the EXTBKUP function, one or more FILEOUT functions must have been specified. FILEOUT functions define which tables are extracted by the EXTBKUP function. See FILEOUT (Define Output Data Sets).

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 generates data in expanded form. An extract file is not compressed and contains no CA Datacom control information. However, 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.

The FILEOUT function (see FILEOUT (Define Output Data Sets)) 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.

The z/OS input concatenation of multiple individual backup files is supported by EXTBKUP.

Prior to the EXTBKUP function, one or more FILEOUT functions must have been specified to define which tables are extracted.

When to Use

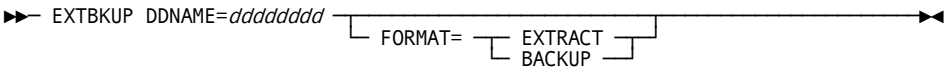
Use the EXTBKUP function when you intend to process table information for use with the CA Datacom Datadictionary Record Migration Facility utility (DDRMFLM) in fixed-block format or for use outside the CA Datacom environment.

You can also use EXTBKUP FORMAT=BACKUP to create an uncompressed copy of the backup file.

How to Use

The MUF must be active when you execute this command. Execute the EXTBKUP function using the following command format:

Extract from Backup File (z/OS)



Command

EXTBKUP

Invokes the function to extract data from a backup file in expanded form.

Required Keywords

DDNAME=

Specifies the name of the input backup file.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to z/OS JCL specifications

Default Value:

(No default)

Optional Keywords

FORMAT=

If you do not specify FORMAT= (or if you specify FORMAT=EXTRACT), output is one or more extracted files, as was the case before Version 12.0 (in r11). 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.

```
//jobname    See the note above and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=4M
//STEPLIB    See the note above and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//POLOUT   DD DSN=db001.pol,DISP=SHR             Output extract file
//INFILE   DD DSN=infile,DISP=SHR                Input data set
//SYSIN    DD *                                  Command Input
         FILEOUT  DDNAME=POLOUT,TABLE=POL
         EXTBKUP  DDNAME=INFILE
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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 455)) 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.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
FILEOUT DDNAME=POLOUT,TABLE=POL

FUNCTION=FILEOUT
   DDNAME=POLOUT
   TABLE=POL
```

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.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
EXTBKUP DDNAME=INFILE

FUNCTION=EXTBKUP
   DDNAME=INFILE
```

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.

```
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
```

This page of the report shows the following:

Confirmation that records were extracted from the requested table, as specified by the FILEOUT statement.

Chapter 19: 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*.

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.

In Simplify mode, the function expects MUF to be enabled. This provides control and protection. If MUF is not enabled, this function should wait for MUF to enable. If you have to execute the function without MUF enabled, for example if it is needed to allow restart to execute, the function must either follow a function that can only run with MUF not enabled or follow a function to acknowledge MUF is not enabled with SET OPTION1=MUF_NOT_ENABLED. It is possible (but not recommended) to use SET OPTION1=MUF_ENABLED_OR_DISABLED. The function without MULTUSE=YES and MUF enabled does not require or want the database to be closed but there must be no open URT for any table in the base.

How Space Is Allocated for an EXTEND

The additional space into which a database data or index area is to be extended can be allocated in one of four ways:

1. Dynamic allocation is always allowed.

   If you specified a secondary allocation parameter when you originally allocated the data set containing the area, you need not supply an z/OS JCL DD statement when running EXTEND (unless a new volume is being added).

   As long as all secondary allocations for the data set have not already been used and sufficient space exists on the last volume allocated to the area to contain the secondary allocation, the EXTEND function automatically requests another extent that is the size of the original secondary allocation specification. The function initializes this new space and resets the control record for the area accordingly.

2. You can supply an z/OS JCL DD statement with a disposition (DISP=) of SHR or OLD and include a SPACE parameter with a secondary allocation amount. As in the previous case, the EXTEND function causes the allocation of this secondary space amount (provided all secondary allocations have not been exhausted), initializes the new space, and resets the control record.

3. You can cause the EXTEND function to add another disk volume to the data set, allocate additional space on the new volume, and initialize the new space. To do this, you must specify an additional disk volume serial number in the z/OS JCL DD statement.

   Also, there must be less space available on the last volume currently allocated to the data set containing the area than you specify for the secondary allocation. This forces allocation of the next extent on the new volume.

   To use this method:

   a. Uncatalog the data set manually (for example, using IEFBR14).

   b. Extend the data set and specify the following in the JCL:

      - Code a SPACE parameter in the DD statement for the area that has a primary quantity equal to the initial primary allocation amount and specify the secondary quantity with the amount by which to extend the data set.
      - For the VOL=SER information, specify the volumes in the same order by which the file was allocated and extended.
      - With the new volume serial number, specify DISP=(OLD,CATLG).

4. You can move the data or index area to another larger data set on another volume of the same device type. You can do this using a non-CA Datacom/DB utility.

   In this case, the EXTEND function compares information from the IBM Data Extent Block (DEB) to the control record information. EXTEND then formats any new blocks based on the extent information from the DEB. The function updates the control record to reflect the new block count.

Verifying the Extend

Use the CXX Report Type A (see Type A - Directory Space Utilization) to verify that the total number of tracks match what the VTOC has for all volumes on which the data set is placed. If the number of tracks are wrong, restore the data set.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO, Simplify NO, or MUF down*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any MUF for read.
- The database may *not* have any table in *unloading* status.

Environmental **Requirements** *when MULTUSE=NO, Simplify YES, and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when MULTUSE=NO, Simplify NO, or MUF down*

- 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 **Controls** *when MULTUSE=NO Simplify YES and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

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

- The database may or may not be open for update in the connected MUF.
- The database may *not* be open for update in another MUF 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.

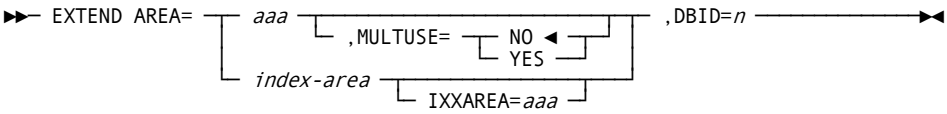
- The area may *not* have any other MULTUSE=YES utility executing.
- The area must be closed.
- When MULTUSE=YES, this function will not be allowed against a Data Sharing MUF if more than one is executing.

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

- The database will be opened by MUF for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets EXTEND executing this MUF this database or area.

How to Use

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 MUF and Single User modes. The database involved must be able to be opened for update in the MUF 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 MUF. The actual data set open is performed in the utility region, and the JCL requirements are applied to the utility region.

Note: The operating system performs the catalog portion of the DD statement DISP option when the DD statement is removed. This occurs at job step termination. Therefore, if you were to change the volume list during an extend, do not expect the MUF to correctly open this area until after the step completes. This restricts the DBUTLTY functions that succeed when following a catalog change.

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 about area level DBUTLTY control, see Area Level DBUTLTY Control.

Valid Entries:

NO or YES

Default Value:

NO

IXXAREA=*aaa*

The IXXAREA= keyword is optional when AREA=IXX is also specified and 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)

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.

```
//jobname    See the note above and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the note above and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//SYSIN    DD *                                  Command Input
         EXTEND  AREA=DEM,DBID=1
/*
```

In the DEM001 statement above, the space parameter is not required if the DEM area was previously allocated with secondary extents. It is required when changing the secondary value or when a new volume is added.

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
EXTEND AREA=DEM,DBID=1

FUNCTION=EXTEND
   AREA=DEM
   DBID=1
```

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.

```
DEM - STATUS INFORMATION
DEVICE TYPE -    3380
BLOCK LENGTH -  1,024
BLOCKS/TRACK -     31

BLOCKS IN USE -        13
BLOCKS UNUSED -       142
TOTAL BLOCKS -        155
TOTAL TRACKS -          5
PERCENT FULL -          9
```

This page of the report shows the following:

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 20: 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.

More Information:

- For more information about DBUTLTY STATUS console command support for this function, see COMM STATUS (Print Status of MUF Jobs) (see page 283).
- For more information about the impact of table partitioning on the EXTRACT function, see The EXTRACT Utility**.**

When to Use

Use the EXTRACT function when you intend to process table information for use with the CA Datacom Datadictionary 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.

In Simplify mode, the function expects MUF to be enabled. This provides control and protection. If MUF is not enabled, this function should wait for MUF to enable. If you are required to execute the function without MUF enabled, for example if it is needed to allow restart to execute, the function must either follow a function that can only run with MUF not enabled or follow a function to acknowledge MUF is not enabled with SET OPTION1=MUF_NOT_ENABLED. It is possible (but not recommended) to use SET OPTION1=MUF_ENABLED_OR_DISABLED. The function without MULTUSE=YES and MUF enabled does not require or want the database to be closed, but there must be no open URT for any table in the base, except that SEQ=PHYSICAL allows URT entries.

Extracting in Native Key Sequence

When not using Simplify or MUF is not enabled, 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 helps ensure 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 help 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 MUF 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.

When using Native Key sequence you can do a parallel extract. See Parallel Extract (see page 451).

Extracting in Physical Key Sequence

With UPDATE=YES, the physical extract occurs with the rules as previously described (for Native Key sequence).

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.

Use the EXTRACT function to verify the integrity of every data block in a data area. To activate this feature, code 'DUMMY' for the DDNAME= keyword value (DDNAME=DUMMY) and also code 'DUMMY' on the JCL DD statement.

```
//DUMMY DD DUMMY
```

If any block in the data area is determined to be invalid, the DBUTLTY execution terminates with a S000 U004 abend. Additionally, it produces one DB13155W message for each invalid data block found. SNAP dumps are produced for the first ten errors encountered.

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, Simplify NO, or MUF down*

- The database may not be open for update anywhere.
- The database may or may not be open in any MUF for read.
- The database may or may not have any table in unloading status.

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

- The database may be open for update but must have no users.

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

- 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 **Controls** *when MULTUSE=NO and either SEQ=NAT or SEQ=PHY,UPDATE=YES, Simplify YES, and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

Environmental **Requirements** *when SEQ=PHY,UPDATE=NO, Simplify NO, or MUF down*

- 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 SEQ=PHY,UPDATE=NO, Simplify YES, and MUF enabled*

- The database may be open for update or will be opened.

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

- The database may or may not be open for update in the connected MUF
- The database may *not* be open for update in another MUF 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 will not be allowed against a Data Sharing MUF if more than one is executing.

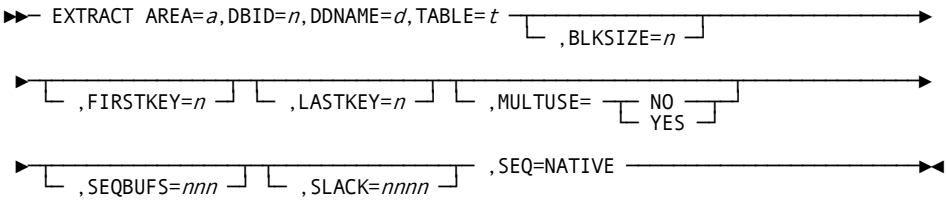
Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by MUF for update.
- The database will have no *unloading* status set.
- The utility sets ACCESS area READ (utility set).
- The utility sets EXTRACT executing this MUF this database or area.

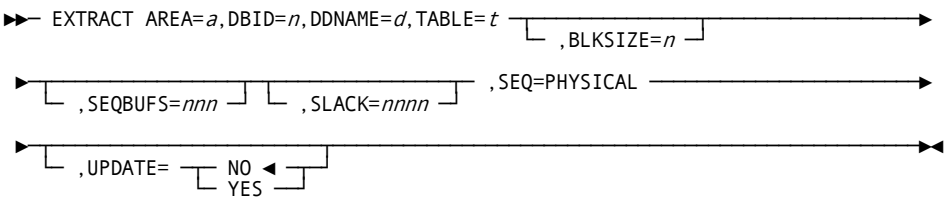
How to Use

You can execute this command in either Single User or with the MUF active. Execute the EXTRACT function using the following command format:

Extract Data Table Native (z/OS)



Extract Data Table Physical (z/OS)



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)

,DDNAME=

Specifies the JCL DDNAME for the output data set. This name must match the corresponding name in the JCL.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to z/OS JCL specifications

Note: As previously mentioned, the name cannot be the standard DDNAME 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:

The default for BLKSIZE= when running z/OS is 0 (zero), which defaults to the z/OS default. That default is large and optimized for the particular device type, that is, the most optimal default for sites executing this operating system.

,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 utility 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.

,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 about area level DBUTLTY control, see Area Level DBUTLTY Control.

Valid Entries:

NO or YES

Default Value:

NO

,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. One of the uses of FIRSTKEY=/LASTKEY= involves doing parallel extracts. For more information, see Parallel Extract (see page 451).

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=

With MULTUSE=YES specified, if necessary to ensure best performance DBUTLTY overrides the SEQBUFS= value either specified or defaulted. If SEQBUFS= is omitted (invoking the default value) or specified as a value 0-128 (even numbers only), the MUF treats this as though 128 was specified (or defaulted). This directs the MUF to allocate 128 private sequential buffers for use by this function. If SEQBUFS= is specified as a number 130-256 (even numbers only) the MUF treats this as though 256 was specified and uses the common data pools (64 blocks per I/O).

When MULTUSE= is omitted or specified as NO, SEQBUFS= is edited and ignored, and DBUTLTY executes with the best performance by using either the available data buffers (specified with the DATANO= parameter in the DBMSTLST macro) or available private sequential buffers.

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

Parallel Extract

Use parallel extract by specifying starting and/or ending key value ranges to be processed. Using parallel extract you could, for example, extract the first half of the data in one job and the remaining half in a concurrently executing second job. CA Datacom/DB places no restrictions on the number of extracts running in parallel. We recommend a starting point of four so that each processes approximately one fourth of the data. It is up to you to determine the appropriate key value ranges.

The parallel extract output file can be used as input to a regular LOAD DBUTLTY function either as all parallel parts concatenated together in key value order (for a full load) or less than all of the parallel parts to allow for deleting a range of rows.

FIRSTKEY= and LASTKEY= Keywords in Parallel Extract

It is important that the set of parallel extracts not miss any data rows, and *it is your responsibility* to edit the key values before executing the set of extract jobs. We strongly recommend that each set of extract jobs be executed through the DBUTLTY edit function and the output reviewed before execution. To do the edit, add the parameter EDIT as the first DBUTLTY function followed by your parallel extract control statements. Execute DBUTLTY and review the output to make certain that you are requesting what you expect.

If the parallel extract is being used to delete from the start or end of the key value range, the extract can accomplish this with one job step. A load of the output extract file would provide a data area loaded without the *dropped* rows.

When DBUTLTY echoes back the input parameters for the FIRSTKEY= and LASTKEY= keywords, it prints them as a three line set with the first line being printable characters, the second being the zone portion of the bytes, the third being the digit portion of the bytes. Following is an example:

```
EXTRACT DBID=111,AREA=SAM,DDNAME=EXTRACT,SEQ=NATIVE
 FIRSTKEY=0512345,LASTKEY=0299

FUNCTION=EXTRACT
   AREA=SAM
   DBID=00111
   DDNAME=EXTRACT
   SEQ=NATIVE
   FIRSTKEY=0512345
   . . .  .=FFFFFFF
   . . .  .=0512345
   LASTKEY=0299
   . .  . =FFFF
   . .  . =0299
```

The following is an example of parallel extract using three output datasets:

```
   EXTRACT DBID=790,AREA=A01,DDNAME=EXTRACT1,SEQ=NATIVE,
      FIRSTKEY=00,LASTKEY=043333

   EXTRACT DBID=790,AREA=A01,DDNAME=EXTRACT2,SEQ=NATIVE,
      FIRSTKEY=043334,LASTKEY=046666

   EXTRACT DBID=790,AREA=A01,DDNAME=EXTRACT3,SEQ=NATIVE,
      FIRSTKEY=046667,LASTKEY=00
```

Restrictions

We do *not* recommend parallel extract for tables with a NATIVE sequence key defined as descending or as being type sensitive. For these types of keys, the actual key value stored in the index may differ from the value stored in the data row. Parallel extract is therefore allowed with the restriction that the key value provided must be in the internal index format, not in the external format matching the data row data.

Other restrictions for parallel extract are as follows:

- The option is available only in z/OS.
- The option is available only to EXTRACT with SEQ=NATIVE and AREA= specified.
- Exercise due caution when specifying and using parallel extract. Be aware that data rows could be lost due to poorly defined control statements or data set handling. CA Datacom/DB cannot edit the goal, only the specific control statements provided.

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.

The CA Datacom/DB SVC must be installed when using multivolume data sets. See the *CA Datacom/DB Database and System Administration Guide* for further details.

Example JCL

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 or disk. 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.

```
//jobname    See the note above and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the note above and JCL Requirements.
//CXX      DD DSN=cxx.data.set,DISP=SHR          Directory data set
//OUT1DD   DD DSN=out1dd,DISP=SHR                Output data set
//SYSIN    DD *                                  Command Input
         EXTRACT  AREA=DEM,DBID=1,DDNAME=OUT1DD,TABLE=POL,SEQ=PHYSICAL
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
EXTRACT AREA=DEM,DBID=1,DDNAME=OUT1DD,TABLE=POL,SEQ=PHYSICAL

FUNCTION=EXTRACT
   AREA=DEM
   DBID=1
   DDNAME=OUT1DD
   SEQ=PHYSICAL
   TABLE=POL
```

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.

```
 TABLE          RECORDS

 PNC                 0
 PNM                 0
 POH                 0
 POL                 4

 TOTAL               4
```

This page of the report shows the following:

Confirmation that four records were extracted from the requested table.

Chapter 21: 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)). If the table is empty, FILEOUT creates a valid null output data set.

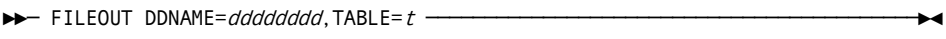
Note: When used to define one or more output data sets used by the EXTBKUP function, all FILEOUT statements must precede the EXTBKUP statement.

When to Use

Use the FILEOUT function when you want to define output data sets.

How to Use

The MUF must be active when you execute this command. Execute the FILEOUT function using the following command format:



Command

FILEOUT

Invokes the function to define output data sets.

Required Keywords

DDNAME=

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.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

A valid extract output file name.

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)

Examples

For sample JCL showing the use of the FILEOUT statement in the EXTBKUP function, see Example JCL for EXTBKUP (see page 430).

For a sample EXTBKUP report showing the use of the FILEOUT statement, see Sample Report: EXTBKUP.

Chapter 22: 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 MUF 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 MUF startup option and the DBUTLTY function (and the DATAPOOL or SYSPOOL buffers) of 99999 for each class. The alternate buffer pools defined with BUFFER_POOL_DEF also count in the maximum per class (IXX, DXX, data).

The FLEXPOOL function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, help ensure that the TOGROUP DBSIDPR parameter is correctly defined. For more information about XCF, see the *CA Datacom/DB Database and System Administration Guide*.

When using a console command, these steps are unnecessary.

When to Use

Use FLEXPOOL to add or delete flexible buffers.

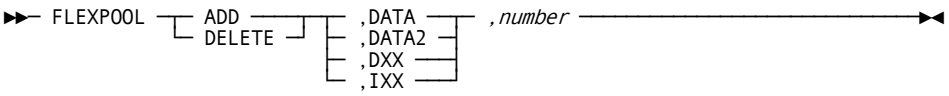
To see the current flexible buffer counts, use a FLEXPOOL add or delete statement for the desired buffer class with 0 specified. The function generates a message, on the console or on the DBUTLTY report, indicating the status of the request and the number of buffers added or deleted, remaining, and totals.

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

The MUF must be active when you 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:

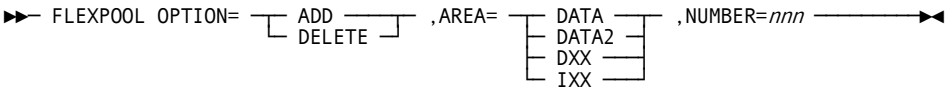
0 to 9999

Default Value:

(No default)

How to Use the DBUTLTY Command

The MUF must be active when you 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
         FLEXPOOL  OPTION=ADD,AREA=DATA,NUMBER=20
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
FLEXPOOL OPTION=ADD,AREA=DATA,NUMBER=20

FUNCTION=FLEXPOOL
   OPTION=ADD
   AREA=DATA
   NUMBER=20
```

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.

```
DB00606I - FLEXPOOL ADD    DATA 00020, NOW TOTAL 00220 FLEXPOOL 00020
```

This page of the report shows the following

FLEXPOOL ADD

Indicates the ADD or DELETE function.

DATA

Indicates the buffer class (data, IXX, or DXX).

00020

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.

TOTAL 00220

Indicates the total number of buffers in this class (includes the MUF startup options, DATAPOOL and SYSPOOL, plus any FLEXPOOL buffers).

FLEXPOOL 00020

Chapter 23: FORCE (Force Keyword Values)

By making use of the ability of the DBIN1PR Macro to process user configuration options prior to normal SYSIN processing, the DBUTLTY FORCE function allows you to force an allowed keyword value.

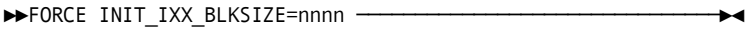
Important! We recommend that you use the DBUTLTY FORCE function with caution. We recommend caution because when most users note a condition code of 0 (zero), they do not additionally view output. Therefore, other than expected results are not noticed. For that reason, it is simpler and of less potential confusion to all DBUTLTY users if the actual JCL streams are adjusted for site standards, instead of making use of the FORCE function.

When to Use

Use the FORCE function when you want to force the allowed keyword values.

How to Use

Use an alternate (additional) SYSIN with a DBUTLTY FORCE function to force a keyword value and ignore a different user specification. That is, using FORCE forces the value you specify to be used, preventing other users from performing any overrides.



Command

FORCE

Invokes the function to force certain keyword values.

Use no blank before FORCE when you specify it in columns 1-7. Follow FORCE with one blank (and use no continue lines).

Required Keyword

INIT_IXX_BLKSIZE=*nnnn*

INIT_IXX_BLKSIZE=*nnnn* gives you the ability to force a value *nnnn* for the block size during an INIT function with AREA=IXX used to initialize an index area (IXX or multiple data set index).

Example

Suppose an alternate SYSIN contained the following statement:
FORCE INIT_IXX_BLKSIZE=5555

With that statement, 5555 becomes the block size used for an INIT of an index area, even if the non-alternate SYSIN contains a different BLKSIZE value.

Consider the following:

- The alternate (additional) SYSIN can be overridden for any step by having a JOB/STEPLIB pointing to a different DBIN1PR module.
- If the DBIN1PR module indicates a data set, the additional SYSIN can be overridden for any step by having a DD statement pointing to a different input.
- All DBIN1PR input is echoed to the console without option, for example: DB10096I - DBIN1PR PROCESSED - FORCE INIT_IXX_BLKSIZE=5555.
- Even with the use of the FORCE function, you have to enter a valid value if you specify the option.

  For example if you set a keyword with an invalid value of BLUE (BLKSIZE=BLUE) for an index INIT, that error is flagged and the function fails until the error is removed or corrected. If the value is correct, however, the FORCE is of course appropriately applied.
- The use of FORCE does not force an error.

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    INIT AREA=IXX,DBID=997,BLKSIZE=7777

                   FUNCTION=INIT
                      AREA=IXX
*****- NOTE  -*****   BLKSIZE=05555  DB10029W - DBIN1PR FORCE APPLIED
                      DBID=00997
```

This page of the report shows that, to make the FORCE more visible in the report, a note and message are displayed to indicate that a FORCE has been applied.

Chapter 24: GAPFIX (Fix CDC Gaps)

The GAPFIX function allows you to fix gaps that occurred during Change Data Capture (CDC) processing. Using CDC does not normally cause any processing gaps. However, you need to know what action to take when gaps do occasionally occur.

Note: For more information about CDC, see the *CA Datacom/DB Database and System Administration Guide*.

When to Use

Use the GAPFIX function when a gap occurs during Change Data Capture (CDC) processing.

A gap can occur at the following three points in time:

- The first gap can occur when a MUF is defined to have CDC set for a set of databases, but the MUF is then enabled and runs for an extended period of time without an established or started Change Data Capture Listener (CDCL) task, during which time the LXX spills its first CDC record. The CDCL task fails to see the spilled data when it is established or started after the spill, because the spilled data is then missing from the LXX. This point in time includes cases in which the CDC database was for some reason null loaded.
- The second gap can occur when the source MUF and the CDCL task are running normally, the CDC database becomes full, and time passes before the full database is noticed and resolved. By the time the full database is resolved, the LXX no longer has enough room available for the work that occurred before the resolution took place, resulting in some of the CDC data being spilled and overlaid on the LXX. When the CDCL task is restarted, the gap is detected.
- The third gap can occur when the CDCL task starts and processes prior CDC records (the prior records are known to the associated control record) but then discovers that while it was stopped the LXX was initialized or reset. This situation indicates that data could have existed while the CDCL task was stopped but before the LXX was initialized or reset. In this case, provide that missing data before you start new CDC records.

You handle each of the three previously described cases based on whether the control record still exists.

- If the control record still exists, execute the GAPFIX function of DBUTLTY with the ordered RXX file or files from the date-time of the gap. The date-time of the gap is provided in message DB03121I. GAPFIX directs message DB03121I to the MUF JES log. The STARTED and ENDED examples of message DB03121I are:

  ```
  DB03121I - DBCPGPR DBDVL A-DBDVM01 STARTED LASTLXX-00000000000000

  DB03121I - DBCPGPR DBDVL A-DBDVM01 ENDED LASTLXX-20060622142058
  ```

  The date-time of the gap is provided in the ENDED form of message DB03121I following the LASTLXX in the format of ccyymmddhhmmss..

  - cc is the century
  - yy is the year
  - mm is the month
  - dd is the day
  - hhmmss to give the hour and minutes and seconds.

  For more information about message DB03121I, see the *CA Datacom/DB Message Reference Guide*.

  Note: GAPFIX also can generate error message DB03122E and informational message DB03123I.

  For more information about messages DB03122E and DB03123I, see the *CA Datacom/DB Message Reference Guide*. Examples are:

  ```
  DB03122E - DBCPGPR ERROR RXX EMPTY

  DB03123I - DBCPGPR ENDING
  ```

  The GAPFIX function reads the control record and processes the RXX data in the same way that the CDCL task processes the LXX directly. The GAPFIX function adds TSN records and adds or deletes MNT records, just as CDCL does. It also updates the control record and does commits, similar to CDCL, but because the RXX is not track-oriented, the commits occur either after every 1000 CDC records or 10000 log records. The GAPFIX function of DBUTLTY can always be restarted if for some reason it fails.

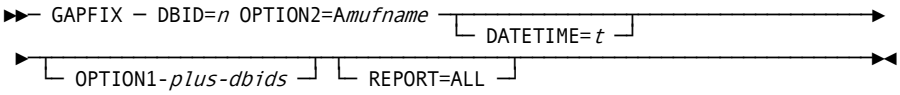
- If the control record existed and you provided the starting date and time, if a commit has occurred a control record now exists and the date and time option must be removed when running the GAPFIX function. All RXX data needs to be processed, or at least all not currently on the LXX. If the CDCL task is set to open, the last log record handled by the GAPFIX function remains on the LXX and can therefore restart with no risk of missing data. Data spilled to the LXX still exists on the LXX until it has been overlaid with new data.

If the control record does not exist, it might be because CDC is just starting or because the CDC database has been lost. If a control record does not exist, use GAPFIX and begin by choosing a starting date and time to use as the value of the GAPFIX DATETIME= keyword (see How to Use). In executing the GAPFIX function, ensure that the first and all subsequent RXX files after the date and time you specify are included. When GAPFIX execution begins, it makes certain that no control record exists then adds a record with a starting date and time of the first log record that matches or comes after the DATETIME= you specified. GAPFIX processes all RXX data in the same way as CDCL would. At completion, the CDC control record is updated to reflect the last RXX log record. After the GAPFIX function completes, you have two choices as follows:

- You can run GAPFIX again, providing the last RXX file and newer RXX files, if any exist.
- Alternately, you can direct the CDCL task to open with CDCL_OPEN, causing the CDCL task to open and read the control record. If the oldest log record processed is now on the LXX, the CDCL task continues with normal processing. If the LXX does not have the needed record, it indicates that a new gap has been found. The CDCL task issues a new gap message, and closes.

How to Use

The MUF must be active when you execute this command. To fix a gap that occurred during CDC processing, use the following command.



Command

GAPFIX

Invokes the GAPFIX function. GAPFIX must start with a proper file boundary, not within volumes. Because GAPFIX reads and updates the control record, it cannot run if the CDCL task is open. If the RXX contains no records, an error is recognized and GAPFIX fails.

Required Keywords

DBID=*n*

Identifies the CDC database, where *n* is the CDC database ID. This keyword allows you, if necessary, to specify a system database ID for the CDC database other than the default DBID of 2009.

Valid Entries:

Valid database ID of the CDC database

Default Value:

2009

OPTION2=A*mufname*

Identifies the letter A as the control ID, and identifies the MUF or MUFplex in the *mufname.*

The control ID occurs in every CDC database record that is built and defines record format mapping. It does not indicate a release level.

The MUF or MUFplex name must match that of the RXX file header. If the control record exists, this MUF or MUFplex name must also match that of the control record. The MUF name or MUFplex name used in GAPFIX is known only from the cycle start record.

Valid Entries:

the letter A followed by a valid MUF or MUFplex name

Default Value:

(No default)

Optional Keywords

DATETIME=*t*

DATETIME= specifies the date and time at which you want to start the GAPFIX function.

Valid Entries:

a valid date and time at which to start the GAPFIX function

Default Value:

(No default)

OPTION1=*plus-dbids*

Specify the database IDs for one through eight PLUS databases.

For example, adding one PLUS database with a database ID of 2012 is specified as follows:

```
OPTION1=2012
```

If specifying more than one PLUS database, omitted databases are indicated by a comma, if they are not at the end (termination) of the list. For example:

```
OPTION1='2012,,,,,,,2019'
```

Note: A list of databases is surrounded by single quote marks as shown.

Valid Entries:

1-8 valid 4-digit database IDs

Default Value:

(No default)

REPORT=ALL

Specifying REPORT=ALL generates a detail report that you can use for debugging and testing purposes. The detail report provides a line for every CDC log record that is processed. For a sample detail report, see GAPFIX Detail Report Sample.

Note: The GAPFIX function always prints a non-detail report when it completes. For a sample non-detail report, see GAPFIX Non-Detail Report Sample.

Valid Entries:

ALL

Default Value:

(No default)

GAPFIX Reports

For testing purposes, you need not ever open the CDCL task. Testing can be done with the GAPFIX function after the data is spilled.

GAPFIX has access only to RXX files, not LXX files. GAPFIX cannot validate the LXX INIT/RESET dates, nor can it provide them if a missing control record has been added by the GAPFIX function. The first CDCL task adds the current INIT/RESET dates and presumes the user has not done a reset.

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                                        DATACOM/DB FIXGAP

   TOTAL LOG      CDC LOG        ADDED        ADDED      DELETED
     RECORDS      RECORDS  TSN RECORDS  MNT RECORDS  MNT RECORDS
      49,109       12,222           12       12,001            1

 LOW DATE/TIME             LOW STCK      HIGH DATE/TIME                HIGH STCK
20060622142058 00BEFE47790C696F0800      20060622142101      00BEFE477C25F6D18E00
```

This page of a GAPFIX non-detail report, that always prints when the GAPFIX function completes, displays the following information:

TOTAL LOG RECORDS

Displays the number of log records read starting with the one matching a DATETIME= option or the one following the required matching record when not using DATETIME=. It reflects the records actually processed, not skipped.

CDC LOG RECORDS

Displays the number of log records not skipped, as previously described. This includes records subject to CDC and also commit and backout type records.

ADDED TSN RECORDS

Displays the number of TSN type records added.

ADDED MNT RECORDS

Displays the number of MNT type records that were added.

DELETED MNT RECORDS

Displays the number of MNT type records that were deleted. These occur due to various types of backout conditions. More deletes occur than adds because a previous execution could have added the records with the rollback occurring in that execution.

LOW DATE/TIME and LOW STCK

The LOW DATE/TIME and LOW STCK (low store clock) values represent the lowest record processed, that is, not skipped.

HIGH DATE/TIME and HIGH STCK

The HIGH DATE/TIME and HIGH STCK (high store clock) value is the highest log record processed.

GAPFIX Detail Report Sample

```
•                                        DATACOM/DB FIXGAP
•JOB NAME R-UNIT   DATE       TIME   REQUEST  BASE  BLOCK      URI  AREA PLUS SLOT
•
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITC01  997 00000002 00000001 A01 12------9
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITC02  897 00000002 00000001 C02 12------9
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITF01  797 00000000 00000002 A01 12------9
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITF02  697 00000002 00000001 A01 --------9
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITF03  597 00000002 00000001 A01 -2-------
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITRNA  497 00000002 00000001 A01 1--------
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITRNB  397 00000002 00000001 A01 --------9
•DBDVL13  80,829  8/14/2013 10.21.32 COMIT       0 00000000 00000000
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITC01  997 00000002 00000003 A01 12------9
•DBDVL13  80,829  8/14/2013 10.21.32 ROLBK       0 00000000 00000000
•DBDVL13  80,829  8/14/2013 10.21.32 DELETC01  997 00000002 00000001 A01 12------9
•DBDVL13  80,829  8/14/2013 10.21.32 DELETC02  897 00000002 00000001 C02 12------9
•DBDVL13  80,829  8/14/2013 10.21.32 DELETF01  797 00000000 00000002 A01 12------9
•DBDVL13  80,829  8/14/2013 10.21.32 DELETF02  697 00000002 00000005 A01 --------9
•DBDVL13  80,829  8/14/2013 10.21.32 DELETF03  597 00000002 00000006 A01 -2-------
•DBDVL13  80,829  8/14/2013 10.21.32 DELETRNA  497 00000002 00000001 A01 1--------
•DBDVL13  80,829  8/14/2013 10.21.32 DELETRNB  397 00000002 00000007 A01 --------9
•DBDVL13  80,829  8/14/2013 10.21.32 DELETRNB  397 00000002 00000002 A01 --------9
•DBDVL13  80,829  8/14/2013 10.21.32 COMIT       0 00000000 00000000
•DBDVL13  80,829  8/14/2013 10.21.32 ADDITRNA  497 00000002 00000008 A01 1--------
•DBDVL13  80,829  8/14/2013 10.21.34 COMIT       0 00000000 00000000
•DBDVL13  80,829  8/14/2013 10.21.34 ADDITF02  697 00000002 00000006 A01 --------9
•DBDVL13  80,829  8/14/2013 10.21.37 COMIT       0 00000000 00000000
•DBDVL13  80,829  8/14/2013 10.21.37 ADDITF03  597 00000002 00000007 A01 -2-------
•DBDVL13  80,829  8/14/2013 10.21.41 CLOSE      0 00000000 00000000
•
```

This is the first page of a sample detail report resulting from specifying the GAPFIX function with REPORT=ALL. Most of the columns are self-explanatory. However, the PLUS SLOT column indicates which CDC_PLUS slot the AUD (add it, update, or delete) log record went to. The example provides some records going to two output slots and some to individual slots. The example provides some records going to two output slots and some to individual slots. The report summary also provides detail plus slot information.

```
                                        DATACOM/DB FIXGAP

   TOTAL LOG     CDC LOG         ADDED          ADDED        DELETED
     RECORDS     RECORDS   TSN RECORDS    MNT RECORDS    MNT RECORDS
         317         116            18            165              3

    CDC_PLUS                     ADDED        ADDED      DELETED
        DBID         SLOT  TSN RECORDS  MNT RECORDS  MNT RECORDS
        2009            1            6           50            1
    CDC_PLUS                     ADDED        ADDED      DELETED
        DBID         SLOT  TSN RECORDS  MNT RECORDS  MNT RECORDS
        3009            2            6           47            1
    CDC_PLUS                     ADDED        ADDED      DELETED
        DBID         SLOT  TSN RECORDS  MNT RECORDS  MNT RECORDS
        4009            9            6           68            1

 LOW DATE/TIME             LOW STCK      HIGH DATE/TIME            HIGH STCK
20130814102124 00CBCF95CC5E04CA5B00      20130814102142 00CBCF95DCD67E2C5800
```

This is the second page of a sample detail report resulting from specifying the GAPFIX function with REPORT=ALL.

Chapter 25: INIT CXX/FXX/IXX/LXX/WXX/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 (see ACCESS (Restricting Opens or Maintenance) for details) to prevent opens for a database that contains the index that you want to initialize.

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. Note that a REPORT with MEMORY=MVS reports the CXX names have been used by every MUF enabled on the System since the last IPL.

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.

Directory Requirements

The Directory can exist as one or more extents on one or more volumes according to operating system requirements and maximums. 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 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.

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.

Important! 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 must execute this command in Single User. MUF must not be enabled. With Simplify mode, this is enforced by several protections. Simplify protections include ensuring that the CXX has not been opened by a previous function without the protections. The INIT CXX function must be the first function in an execution that requires the CXX. Because the INIT CXX is required to run Single User, it establishes Single User protections for all later functions in the same execution.

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.

The name can be a single asterisk (*) to request the name be taken from the DBSIDPR module. With Simplify mode, we strongly recommend this, but if not an asterisk the names are compared and the function fails if they mismatch.

It is a 'best practice' to have the CXXNAME be 1-7 characters to allow for it to also be used as the MUFPLEX name and the MUF name be the same with a suffix of, for example, 1 or 2, or A or B.

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.

```
//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
         INIT     AREA=CXX,CXXNAME=PRODCXX,DATACOM=DB
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
INIT AREA=CXX,CXXNAME=PRODCXX,DATACOM=DB

FUNCTION=INIT
   AREA=CXX
   CXXNAME=PRODCXX
   DATACOM=DB
```

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.

```
CXX - STATUS INFORMATION

DEVICE TYPE -    3380
BLOCK LENGTH -  4,096
BLOCKS/TRACK -     10

BLOCKS IN USE -         2
BLOCKS UNUSED -     2,998
TOTAL BLOCKS -      3,000
TOTAL TRACKS -        300
PERCENT FULL -          1
```

This page of the report shows 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).

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 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.

In Simplify mode, the function expects MUF to be enabled. This provides control and protection. If MUF is not enabled, this function should wait for MUF to enable. If you are required to execute the function without MUF enabled, for example if it is needed to allow restart to execute, the function must follow a function that can only run with MUF not enabled or it must follow a function to acknowledge MUF is not enabled, with SET OPTION1=MUF_NOT_ENABLED. It is possible (but we do not recommend) to use SET OPTION1=MUF_ENABLED_OR_DISABLED.

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.

Multivolume INITs

The data area can exist as one or more extents on one or more 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, Simplify NO, or MUF down*

- The database may not be open for update anywhere.
- The database may not be open in any MUF for read.
- The database may not have any table in unloading status.

Environmental **Requirements** *when MULTUSE=NO Simplify YES and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when MULTUSE=NO, Simplify NO, or MUF down*

- 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 **Controls** *when MULTUSE=NO, Simplify YES, and MUF enabled*

- The database will be opened for update if not already.
- Areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

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

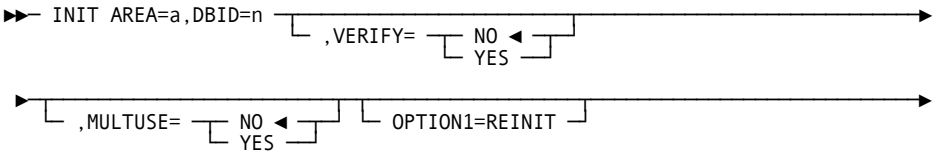
- The database may or may not be open for update in the connected MUF.
- The database may *not* be open for update in another MUF or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set 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 MUF if more than one is executing.

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

- The database will be opened by MUF for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets INIT executing this MUF this database or area.

How to Use

You can execute this command in either Single User or with the MUF active. 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 MUF and Single User modes. The database involved must be able to be opened for update in the MUF 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 MUF. The actual data set open is performed in the utility region, and the JCL requirements are applied to the utility region.

Note: The operating system performs the catalog portion of the DD statement DISP option when the DD statement is removed. This occurs at job step termination. Therefore, if you were to create a new data set with an INIT, do not expect the MUF to correctly open this area until after the step completes. This restricts the DBUTLTY functions that succeed when following a catalog change.

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 about area level DBUTLTY control, see Area Level DBUTLTY Control.

Valid Entries:

NO or YES

Default Value:

NO

,OPTION1=

You can specify REINIT (re-initialize) for the OPTION1= keyword during the initialization of a data area. Specifying OPTION1=REINIT requires that you also specify VERIFY=YES.

When using REINIT, the data set is re-initialized with the same number of tracks that were in use before the INIT, and no additional tracks are found or added to the data set during the INIT. When REINIT is not specified, the INIT closes the data set for update, opens the data set for output (and honors current, complete JCL specifications), closes the data set for output, and then reopens the data set for update.

Adding space to an index or data area is normally accomplished using an EXTEND function, but it can be done during an INIT function as well. A benefit of the OPTION1=REINIT for z/OS users is that REINIT allows unused volumes to be added to data sets that do not have to be removed before an INIT.

For example, if you are not using SMS, you could INIT an area on 3 volumes and load the area. You might then manually uncatalog the area and recatalog it with an additional volume or volumes. Those additional volumes would be available for any dynamic extend for the area that was needed, but to prevent the INIT from forcing the use of the volumes that were added solely for the use of dynamic extend, you would need to manually uncatalog the area before another INIT and recatalog it without any additional volumes.

Valid Entries:

REINIT

Default Value:

(No default)

,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. Note that on some error conditions, the history can be lost.
- 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 inadvertent 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.

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.

```
//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
          INIT     AREA=DEM,DBID=1
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
INIT AREA=DEM,DBID=1

FUNCTION=INIT
   AREA=DEM
   DBID=1
```

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.

```
DEM - STATUS INFORMATION

DEVICE TYPE -    3380
BLOCK LENGTH -  1,024
BLOCKS/TRACK -     31

BLOCKS IN USE -         1
BLOCKS UNUSED -       154
TOTAL BLOCKS -        155
TOTAL TRACKS -          5
PERCENT FULL -          1
```

This page of the report shows messages indicating the 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 How to Use.

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 519) and RETIX (Reconstruct Index) 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 519) and RETIX 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).

In Simplify mode, the function expects MUF to be enabled. This provides control and protection. If MUF is not enabled this function should wait for MUF to enable. If you are required to execute the function without MUF enabled, for example if it is needed to allow restart to execute, the function must follow a function that can only run with MUF not enabled, or it must follow a function to acknowledge MUF is not enabled with SET OPTION1=MUF_NOT_ENABLED. It is possible (but we do not recommend) to use SET OPTION1=MUF_ENABLED_OR_DISABLED.

Successful Execution Requirements and Controls

Environmental **Requirements** *when Simplify NO or MUF down*

- The database may not be open for update anywhere.
- The database may not be open in any MUF for read.
- The database may not have any table in unloading status.

Environmental **Requirements** *when Simplify YES and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when Simplify NO or MUF down*

- The database will be opened for update.

Environmental **Controls** *when Simplify YES and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

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. You can use the ACCESS function (see ACCESS (Restricting Opens or Maintenance) for details) to prevent opens for a database you want to initialize.

Multivolume INITs

The data area can exist as one or more extents on one or more 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 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) and index partition area (Inn) is determined during IXX (or Inn) initialization. When you initialize an index area, 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 index area. Pick an optimal size for your device type, in a manner similar to how the data area block size is selected.

The index block size default provided by CA Datacom can be changed by the DBIN1PR function DEFAULT to be a different value. For details about INIT_IXX_BLKSIZE=, see DEFAULT (see page 405). Any change in default is documented in a console message and in the report, but then allows execution without it being considered an error.

The index block size has the ability to be forced to a specific value even if specified on the function by the DBIN1PR function FORCE to be a different value. For details about INIT_IXX_BLKSIZE=, see FORCE (see page 465). Any force is documented in a console message and in the report, but then allows execution without it being considered an error.

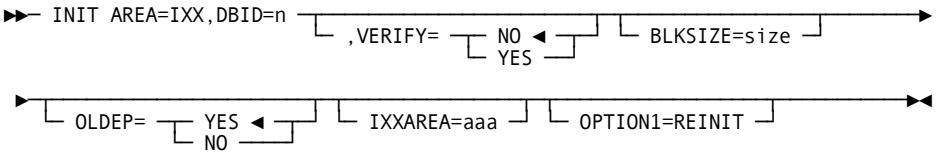
Note: We do not recommend using a block size other than the default. Changing the block size might help in the use of some parts of index storage and processing, but other parts could be hurt by the change. Therefore, changing the block size should only be done after first experimenting to discover whether 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 various 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 use 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. For more information about the *ixxbfrsize* parameter of the SYSPOOL MUF startup option, see *CA Datacom/DB Database and System Administration Guide*.

How to Use

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 inadvertent 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 required keywords in How to Use. However, note that the index block size is subject to a user-provided default that is different than the DBUTLTY default using the DBIN1PR process with function DEFAULT and INIT_IXX_BLKSIZE=. Also note that the index block size is subject to a user-provided force of a specific value, even if BLKSIZE is specified using the DBIN1PR process with function FORCE and INIT_IXX_BLKSIZE= .

We do not recommend changing the block size from the default. For more information, see Controlling the Index Block Size.

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. For more information about the *ixxbfrsize* parameter of the SYSPOOL MUF startup option, see 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

However, note that the YES default value is subject to a user-provided default of INIT_IXX_OLDEP=NO. Therefore, if this option is important, review the output of the execution to see if the user override of the default occurred. For more information see the function DEFAULT (see page 405) and the information about INIT_IXX_OLDEP=NO.

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)

,OPTION1=

You can specify REINIT (re-initialize) for the OPTION1= keyword during the initialization of an index area. Specifying OPTION1=REINIT requires that you also specify VERIFY=YES.

When using REINIT, the data set is re-initialized with the same number of tracks that were in use before the INIT, and no additional tracks are found or added to the data set during the INIT. When REINIT is not specified, the INIT closes the data set for update, opens the data set for output (and honors current, complete JCL specifications), closes the data set for output, and then reopens the data set for update.

Adding space to an index or data area is normally accomplished using an EXTEND function, but it can be done during an INIT function as well. A benefit of the OPTION1=REINIT for z/OS users is that REINIT allows unused volumes to be added to data sets that do not have to be removed before an INIT.

For example, if you are not using SMS, you could INIT an area on 3 volumes and load the area. You might then manually uncatalog the area and recatalog it with an additional volume or volumes. Those additional volumes would be available for any dynamic extend for the area that was needed, but to prevent the INIT from forcing the use of the volumes that were added solely for the use of dynamic extend, you would need to manually uncatalog the area before another INIT and recatalog it without any additional volumes.

Valid Entries:

REINIT

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.

```
//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
          INIT     AREA=IXX,DBID=10
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
INIT AREA=IXX,DBID=10

FUNCTION=INIT
   AREA=IXX
   DBID=10
```

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.

```
IXX - STATUS INFORMATION

DEVICE TYPE -    3380
BLOCK LENGTH -  3,860
BLOCKS/TRACK -     11

BLOCKS IN USE -         2
BLOCKS UNUSED -       108
TOTAL BLOCKS -        110
TOTAL TRACKS -         10
PERCENT FULL -          2
```

This page of the report shows 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 tracks in accordance with CA standards and according to the BLKSIZE= parameter specified. The LXX must contain at least 103 blocks.

For more information about using INIT AREA=LXX to specify the variable Log Area (LXX) block size feature, see Variable Log Area (LXX) Block Size.

When to Use

Use the INIT LXX option to build a new Log Area when creating a new MUF. The Log Area cannot be shared by multiple MUFs. An INIT of the LXX should be a one time occurrence in that version of CA Datacom/DB. If there is ever a need to clear the LXX(LOG), we recommend you perform a DBUTLTY function RESET on the LXX. This clearing of the LXX should never be attempted on a production or development MUF without specific direction from CA Support.

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.

The LXX should never be initialized when it is in use. In Simplify mode, edits protect this requirement in several ways. This function may not execute with the MUF enabled in Simplify mode, and control is established over the CXX to execute without MUF. It may not follow any other functions that use MUF, and none may follow. The function may not follow any function causing the LXX to be opened. To provide better 24x7 support, an exception to some protections exist to allow for a requirement to initialize a new LXX data set intended to replace the current one during a MUF short time outage. The new LXX would have to have a new data set name that would be renamed to the proper name during the MUF outage. To request this special requirement, the INIT AREA=LXX must follow a function of SET OPTION1=SIMPLIFY_BYPASS_EDITS_1.

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

The normal log will be defined as variable, which is described in information that follows. It is rare but possible to define a table with a record size that cannot be supported with the variable format. For this case, the LXX can be set as spanned where a log record can span physical fixed log blocks, it performs slower than variable and is not recommended.

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.

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.

Multivolume INITs

The Log Area can exist as one or more extents on one or more 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.

Variable Log Area (LXX) Block Size

With variable logging, the block size is set to 32760, and MUF uses a variable size to this maximum to ensure most efficient use of DASD. This is based upon the actual log records generated that need to be written to the LXX.

Note:

- 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 MUF startup options: X_LOG_AHEAD_BLKS and X_LOG_MINIMUM_BLK, as described in the *CA Datacom/DB Database and System Administration Guide*.

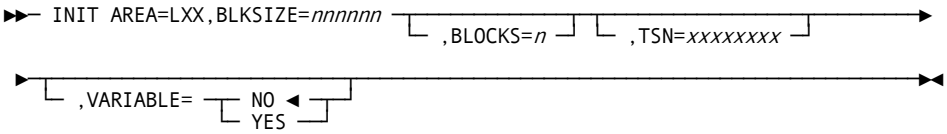
Restrictions and Requirements

Note the following:

- With variable logging, the block size must be set to 32760.
- 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

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. For a variable LXX, the BLKSIZE must be 32760. For spanned logging, DBUTLTY 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.

Valid Entries:

An integer from 16384 to 98280

Default Value:

(No default)

Optional Keywords

,BLOCKS=

For spanned logging, 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 MUF against the same data areas, you should specify a different starting sequence number for each MUF 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). 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 as variable.

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
//LXX      DD DSN=log.data.set,DISP=SHR          Log Area data set
//SYSIN    DD *                                  Command Input
          INIT AREA=LXX,BLKSIZE=32760,VARIABLE=YES
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  INIT AREA=LXX,BLKSIZE=32760,VARIABLE=YES
                   FUNCTION=INIT
                      AREA=LXX
                      BLKSIZE=32760
                      VARIABLE=YES
```

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.

```
LXX - STATUS INFORMATION

                  DEVICE TYPE -    3390
                  BLOCK LENGTH - 32,760
.                 BLOCKS/TRACK -      V - VARIABLE
                  TRACKS IN USE -           2
                  TRACKS UNUSED -       1,258
                  TOTAL TRACKS -        1,260
                  PERCENT FULL -            0
                  CURRENT TSN - 00000000
```

This page of the report shows 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).

Note: 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.

The FXX should never be initialized when it is in use. In Simplify mode, edits protect this requirement in several ways. This function may not execute with the MUF enabled, and control is established over the CXX to execute without MUF. It may not follow any other functions that use MUF, and none may follow. The function may not follow any function causing the LXX to be opened.

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. This INIT of the FXX has to run as a standalone DBUTLTY function for a specific JOB/STEP. You cannot INIT the LXX as part of the same JOB step when attempting to INIT the FXX.

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 except when the LXX is variable, in which case the FXX will not have the VARIABLE=YES option. The number of tracks required is dependent upon the number of tasks generated in the MUF 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 MUF or MUFplex. During initialization (INIT AREA=FXX), the TASKS= keyword is used to specify the number of tasks to be supported. For more information about the TASKS= keyword, see the description in How to Use .

The number of control blocks that are reserved is based upon the following formula: 8192 + (tasks * 64). 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 MUF requires space and space is not available, all log requests fail until the MUF 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.

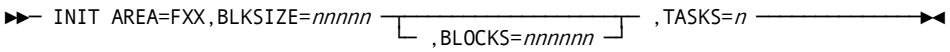
An easy approach is to allow one logical block per task. While active requests are unlikely, the following conditions could increase the probability of their occurring:

- Running in MUFplex mode.
- 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* MUF startup option percentage.
- Having a Log Area (LXX) with relatively few tracks.

How to Use

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. When choosing the number of tasks, consider the following:

- If you are running without CA Datacom STAR in a single MUF, use the number specified for the TASKS MUF startup option. If you have any plans for more tasks in the future, however, specify a larger number now.
- If you are running with CA Datacom STAR in a single MUF, use the TASKS MUF startup option number plus the number of STAR MUF startup option tasks.

- In a Data Sharing environment, do the appropriate previous calculation for *every* MUF in the MUFplex, defined by a specific MUFplex number (1 through 7), that might be enabled or might have been enabled since the LXX was last reset or since the FXX was last initialized. Specifically, if a MUFplex 3 is enabled, space for its tasks must be maintained in the FXX, even if it is ended.

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.

```
//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
//FXX      DD DSN=log.data.set,DISP=SHR          Force Area data set
//SYSIN    DD *                                  Command Input
          INIT     AREA=FXX,BLKSIZE=16384,TASKS=999
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
INIT AREA=FXX,BLKSIZE=16384,TASKS=999

FUNCTION=INIT
   AREA=FXX
   BLKSIZE=16384
   TASKS=999
```

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.

```
 FXX - STATUS INFORMATION

DEVICE TYPE -    3390
BLOCK LENGTH - 16,384
BLOCKS/TRACK -      3

BLOCKS IN USE -         3
BLOCKS UNUSED -       147
TOTAL BLOCKS -        150
TOTAL TRACKS -         50
PERCENT FULL -          2
```

This page of the report displays status information about the Force Area (FXX) at successful completion of the initialization.

INIT WXX (Format RRM Log)

The RRM Log (WXX) is a run-time log used by the Remote Resource Manager (RRM) to resolve outstanding two-phase commits, in conjunction with the Resource Recovery Services (RRS) log. It is maintained by the RRM.

The WXX is not a data set used by a MUF. The INIT may occur with MUF enabled or down, but must be run when RRM is down.

When to Use

Use the INIT AREA=WXX option of the CA Datacom/DB Utility (DBUTLTY) to build a new RRM Log (WXX). You must format the RRM Log with the INIT function before it can be used by CA Datacom/DB.

RRM Log (WXX) Block Size

The WXX should normally be allocated as a very small data set, because it only needs space for outstanding RRM two-phase commits. The space needed can be calculated as follows.

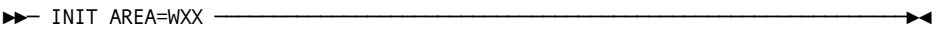
The WXX block size is 4096. The first block is reserved as a control block. Each block has a 32 byte header. Following the header are 96 byte rows, one each per unique outstanding two-phase commit per remote MUF. Each WXX block can contain 42 of these rows.

How to Use

You can execute this command in either Single User or with the MUF active.

Note: The WXX is a small data set. The WXX is required to be a single extent. If the INIT functions finds it to have more than one extent, the INIT function initializes the space then issues message DB02021E and terminates. The use of the WXX by the DBRRMPR execution issues the same message and terminates. The Remote Resource Manager (RRM) only supports a single extent WXX.

Execute the INIT function with the following command format:



Command

INIT

Invokes the format function for CA Datacom/DB use.

Required Keyword

AREA=WXX

Identifies the data set to be formatted as the RRM Log (WXX).

For more information about the Remote Resource Manager (RRM), see the CA Datacom/DB Database and System Administration Guide.

Chapter 26: LINK (Link to the CXX)

When a data area or index area is initialized, it is linked to the Directory (CXX) used in the INIT AREA=CXX function. The link is done using the CXX name. An edit when the data set is opened during all other processing helps ensure that the CXX and data set are linked. If the CXX and data set are found not to be linked, an error is produced.

The LINK function provides protection for all functions except INIT against JCL errors, for example, pointing to the production database instead of the test database. Therefore use extra caution to help ensure that the CXX in use and the data area or index area are correct when submitting an INIT. Of course the edit works when every CXX has a unique CXX name, which we recommend as a best practice. You can run the DBUTLTY REPORT MEMORY=MVS to obtain a list of MUFs on the local LPAR that JOB runs on. This report helps to identify any MUFs that have non-unique CXX names.

When you execute the LINK function, each area in the database is processed in the following sequence:

1. The Index Area (IXX)
2. Other index areas
3. Data areas

Each area is processed as follows:

- The area data set name from the Directory (CXX) is accessed.
- If the data set name is not blanks, the area is opened.
- If the open is successful, the control block of the area is read.
- If the CXX name is found not to be current, it is updated and written.

When to Use

intend to copy a data areaUse the LINK function to link each 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 intend to copy a data area data set from one CXX to another using a utility other than DBUTLTY to backup/load the data area, it is still necessary to do the following:

1. If the receiving CXX data area is set as not loaded, then you need to first use DBUTLTY to INIT and null load the area to a scratch data set to get the area to a loaded status in the receiving CXX.
2. Next, use CXXMAINT OPTION=ALTER.DBID=?.AREA=?.DSN=? to update the data set name to the name you want.
3. Then use the non-DBUTLTY utility to copy the data to the new name.
4. Follow the sequence above by doing the DBUTLTY LINK.
5. Use DBUTLTY RETIX to index the area data that was copied.

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 MUF enable before the LINK function can perform properly. After enabling the MUF, you can perform the LINK function for the Accounting Facility PRM and data area database(s) and then open the desired tables.

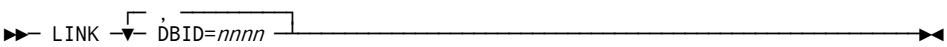
You cannot use the LINK function to fix the Compound Boolean Selection Index Area. You must use the INIT function for the Compound Boolean Selection Index Area. See the *CA Datacom/DB Database and System Administration Guide* for more information.

With Simplify mode, the function can be executed in a DBUTLTY step with the MUF not enabled, for example immediately after a CXX is initialized with a different name and loaded from a backup. If MUF is enabled as is required without Simplify, the other rules apply about databases not being opened before the LINK is executed.

Each area of the database is attempted to be opened, and if the open is successful the LINK performs the action. For each individual area that cannot be opened and read, the action does not occur. A report then reports each area, the actions taken, and any error encountered.

How to Use

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 MUF 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
         LINK     DBID=1-5000
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    LINK DBID=1-5000

                    FUNCTION=LINK
                       DBID=00001-05000
```

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.

```
                                                LINK REPORT
DBID AREA CXXNAME  DSN                                          ERROR INFORMATION

   1 IXX  DBDVM0   ABCDE.DBDVM0.IXX001
   1 DEM
   1 PAY
   1 PMF

   2 IXX  DBDVM0   ABCDE.DB.DBDVM0.IXX002
   2 DD1  DBDVM0   ABCDE.DB.DBDVM0.DD1002

   4 IXX  DBDVM0   ABCDE.DB.DBDVM0.IXX004
   4 PRM  DBDVM0   ABCDE.DB.DBDVM0.PRM004

   5 IXX
   5 A00
   5 A01
   5 A02
   5 A03
   5 A04

   6 IXX

  15 IXX  DBDVM0   ABCDE.DB.DBDVM0.IXX015
  15 DDD  DBDVM0   ABCDE.DB.DBDVM0.DDD015
  15 MSG  DBDVM0   ABCDE.DB.DBDVM0.MSG015
  15 SIT  DBDVM0   ABCDE.DB.DBDVM0.SIT015

  16 IXX
  16 SQ1

  17 IXX
  17 TTM

  97 IXX           X.DBDVM0.IXX097                          74 (81) OPEN ALLOCATION ERROR
  97 A01           X.DBDVM0.A01097                          74 (81) OPEN ALLOCATION ERROR

 297 IXX           X.DBDVM0.IXX297                          74 (81) OPEN ALLOCATION ERROR
 297 I01           X.DBDVM0.I01297                          74 (81) OPEN ALLOCATION ERROR
 297 I02           X.DBDVM0.I02297                          74 (81) OPEN ALLOCATION ERROR
 297 I03           X.DBDVM0.I03297                          74 (81) OPEN ALLOCATION ERROR
 297 I04           X.DBDVM0.I04297                          74 (81) OPEN ALLOCATION ERROR
 297 I05           X.DBDVM0.I05297                          74 (81) OPEN ALLOCATION ERROR
 297 I06           X.DBDVM0.I06297                          74 (81) OPEN ALLOCATION ERROR
 297 I07           X.DBDVM0.I07297                          74 (81) OPEN ALLOCATION ERROR
 297 I08           X.DBDVM0.I08297                          74 (81) OPEN ALLOCATION ERROR
 297 I09           X.DBDVM0.I09297                          74 (81) OPEN ALLOCATION ERROR
 297 A01           X.DBDVM0.A01297                          74 (81) OPEN ALLOCATION ERROR

 397 IXX  DBDVM0   X.DBDVM0.IXX397
 397 A01           X.DBDVM0.A01397                          58 (72) TABLE IS NOT LOADED


 997 IXX  DBDVM0   ABCDE.DB.DBDVM0.IXX997
 997 A01  DBDVM0   ABCDE.DB.DBDVM0.A01997

1000 IXX
1000 SYS

1006 IXX          XXXXO02.DBDVM0.IXX1006                    74 (81) OPEN ALLOCATION ERROR
1006 CBS          XXXXO02.DBDVM0.CBS1006                    74 (81) OPEN ALLOCATION ERROR

1007 IXX          ABCDEF.DB.DBDVM0.IXX1007                  74 (81) OPEN ALLOCATION ERROR
1007 A01          ABCDEF.DB.DBDVM0.A011007                  74 (81) OPEN ALLOCATION ERROR
1007 A02          ABCDEF.DB.MUF0.A021007                    74 (81) OPEN ALLOCATION ERROR


2009 AR1  DBDVM0  ABCDEF.DB.DBDVM0.AR12009
2009 AR2  DBDVM0  ABCDEF.DB.DBDVM0.AR22009

                  *                                     REQUEST COMPLETE
                  DB10045E - ABNORMAL TERMINATION DUE TO ERROR(S)
```

This page of the report shows the following:

If an error prevents the opening of the area or occurs during the I/O events, the error is reported. An area with an error is skipped from having the CXXNAME updated in its control block. Each area successfully processed is reflected in the report. Error messages force the DBUTLTY function to fail, which makes it easy to identify success or failure.

A blank data set name is not reported as an error when:

- A database is defined as VIRTUAL or SYSTEMDBID.
- A data set is in the process of being built and as yet has no data sets.

Return codes of zero (0) are reported so that you can see which have no CXXNAME printed. You can then, if needed, research whether they needed the LINK option.

Chapter 27: 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.

More Information:

- For more information about DBUTLTY STATUS console command support for this function, see COMM STATUS (Print Status of MUF Jobs) (see page 283).
- For more information about the impact of table partitioning on the LOAD function, see The LOAD Utility. FILEIN can be used before a LOAD.
- For more 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.

  Note: 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.

  For CA Datacom Datadictionary base 2 (normally), 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 441) 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 if needed.

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.

Without the Simplify option, verify that the CXX is not open to a MUF or any DBUTLTY function. With Simplify mode, there are edits to help ensure the MUF is not enabled and that the CXX is not in use elsewhere. Also with Simplify, the function either must run as the first function in the step needing the CXX or must follow a function that ensures the CXX was opened with the needed protection. That includes all functions that run only with MUF not enabled or after a function of 'SET OPTION1=MUF_NOT_ENABLED'.

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.
- The Simplify setting is preserved from the current Directory but ignored 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.

- 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 Datadictionary. This catalog can be a direct catalog using CA Datacom Datadictionary online or batch CATALOG or an indirect catalog using CA Datacom Datadictionary 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).

Loading a single database definition with the same or a different database ID is possible. Verify 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 the DELETE option of the CXXMAINT function to delete the database (see CXXMAINT DELETE (Delete CXX Definitions) for details),
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.* 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.

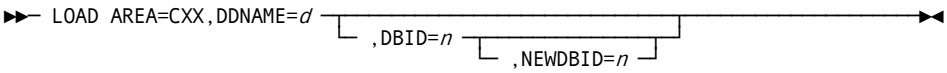
If the database currently exists in the Directory, CA Datacom/DB verifies that the update and read-only counts are zero and the date/time of the last maintenance is the same in the Directory and on the backup.

Without Simplify mode, the load of one or more databases is always performed using the MUF, which must be enabled and the database closed. This is also the expected way to execute with Simplify mode. However, with Simplify mode it is possible for the LOAD of a database to occur with MUF down if the function follows a function that executes only with MUF down and therefore provides CXX protection, or it must occur after you define the MUF to be down using the function SET OPTION1=MUF_NOT_ENABLED. Possible (but we do not recommend) is function SET OPTION1=MUF_ENABLED_OR_DISABLED.

How to Use

MUF execution ensures that the MUF is always in control and has knowledge of all information in the CXX (a normal case being a single MUF or MUFplex) 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. If the DBID parameter is not specified, the entire Directory is loaded.

,DDNAME=

Specifies the JCL DDname for the input data set.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to 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)

,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 Datadictionary 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)

Example JCL

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.

```
//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
//CXXIN    DD DSN=cxx.bkup,DISP=OLD              Directory Backup
//SYSIN    DD *                                  Command Input
        LOAD  AREA=CXX,DDNAME=CXXIN
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    LOAD AREA=CXX,DDNAME=X

                    FUNCTION=LOAD
                       AREA=CXX
                       DDNAME=X
```

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.

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  DATACOM  VERSION
DBDVL      39,237  2011/10/12 04.52.42  DBDVM0           DB       14.0

                    BASE PROCESSED - 1
                    BASE PROCESSED - 2
                    BASE PROCESSED - 3
                    BASE PROCESSED - 4
                    BASE PROCESSED - 5
                    BASE PROCESSED - 6
                    BASE PROCESSED - 10
                    BASE PROCESSED - 15
                    BASE PROCESSED - 16
                    BASE PROCESSED - 17
                    BASE PROCESSED - 97
                    BASE PROCESSED - 197
                    BASE PROCESSED - 297
                    BASE PROCESSED - 312
                    BASE PROCESSED - 397
                    BASE PROCESSED - 497
                    BASE PROCESSED - 597
                    BASE PROCESSED - 697
                    BASE PROCESSED - 797
                    BASE PROCESSED - 897
                    BASE PROCESSED - 990
                    BASE PROCESSED - 997
                    BASE PROCESSED - 1,000
                    BASE PROCESSED - 1,006
                    BASE PROCESSED - 1,007
                    BASE PROCESSED - 1,018
                    BASE PROCESSED - 1,019
                    BASE PROCESSED - 1,020
                    BASE PROCESSED - 1,397
                    BASE PROCESSED - 1,497
                    BASE PROCESSED - 2,009
                    BASE PROCESSED - 5,000
                    *                                        REQUEST COMPLETE
```

This page of the report shows what was loaded and a 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.

  Note: 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 441) 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.

Note: 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 Datadictionary 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 and RECOVERY (Rebuild a Database) (see page 591) 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 Datadictionary 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 when the EXTRACT function of DBUTLTY is used. See EXTRACT (Extract Data Table) (see page 441) for more information. Indicate either form of input by using the FORMAT=EXTRACT parameter.

VSAM Input

CA Datacom/DB accepts VSAM input during the LOAD process. CA Datacom/DB processes the data set specified in the DD statement as it would any data set.

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. See CONFIRM (Verify Constraint Integrity) for 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 Datadictionary 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. See CXXMAINT ALTER DBCS (Activating DBCS Support) (see page 331) for more information.

Force Default Fields

Fields which are defined through CA Datacom Datadictionary 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, concatenate the tables to create a single input file for the final LOAD and indicate FORMAT=BACKUP.
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) for more details. Using this combination eliminates having to back up each table individually and having to concatenate the tables to create a single input source.

LOAD/REPLACE Method

1. Load the first table in the area using the LOAD function.
2. Use the REPLACE function to load the other tables in the area. For details, see REPLACE (see page 641). Using this combination eliminates having to back up each table individually and having to concatenate 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 Not Using REPLACE

To add a table to an existing area that 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. Concatenate the two BACKUP files into one LOAD operation.
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, Simplify NO, or MUF down*

- The database may not be open for update anywhere.
- The database may not be open in any MUF for read.
- The database may not have any table in unloading status.

Environmental **Requirements** *when MULTUSE=NO, Simplify YES, and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when MULTUSE=NO, Simplify NO or MUF down*

- 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 **Controls** *when MULTUSE=NO, Simplify YES and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

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

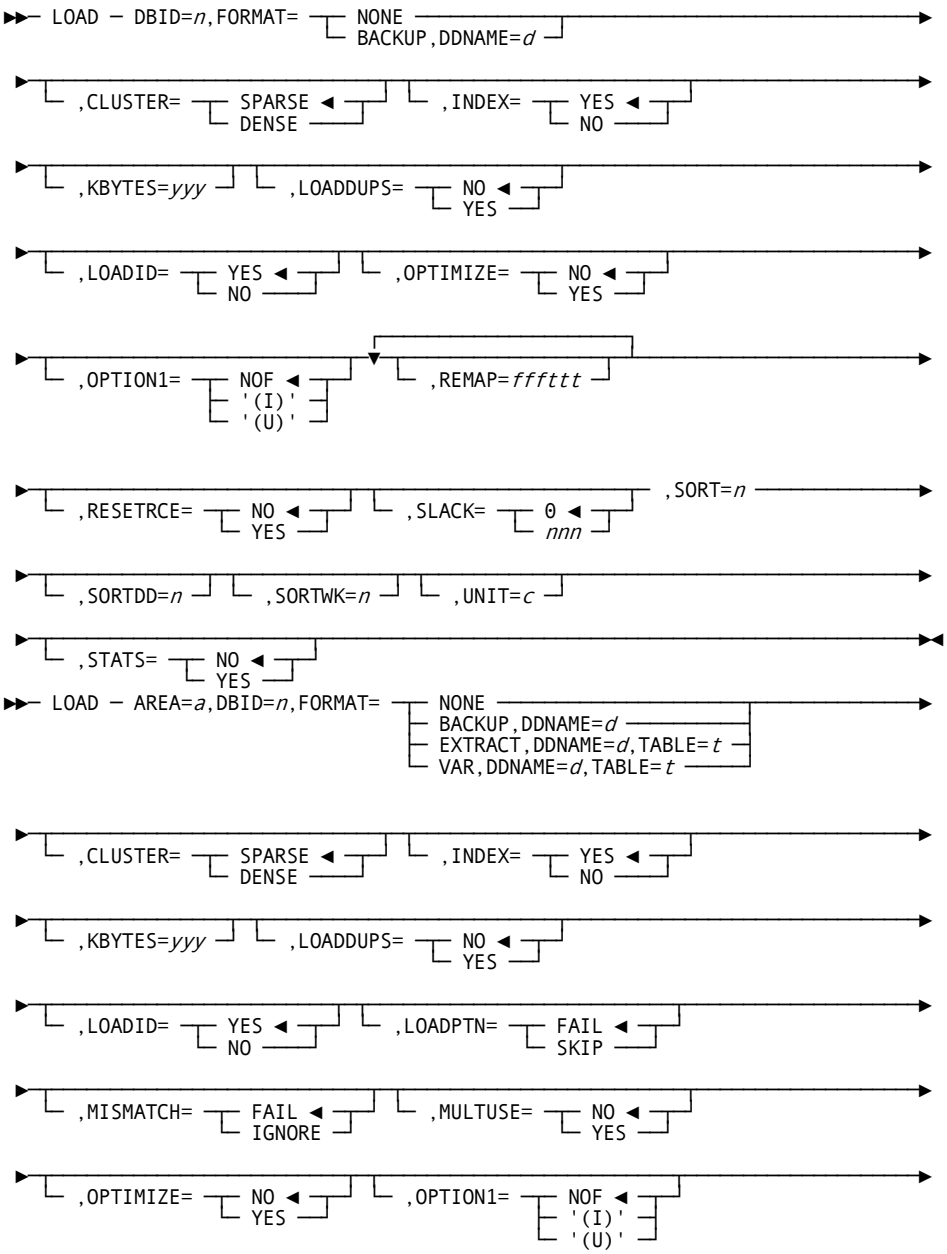
- The database may or may not be open for update in the connected MUF.
- The database may not be open for update in another MUF or Single User job.
- The database may not have any table in unloading status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY
- 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
- When MULTUSE=YES, this function will not be allowed against a Data Sharing MUF if more than one is executing.

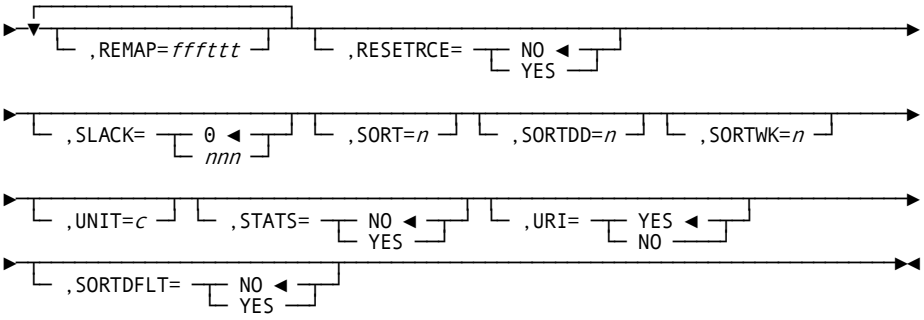
Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by MUF for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets LOAD executing this MUF this database or area.

How to Use

You can execute this command in either Single User or with the MUF active. Execute the LOAD function using the following command format.





Command

LOAD

Invokes the function to build or restore an area.

Required Keywords

DBID=

Identifies the database containing the area to be loaded. If you omit the optional AREA= keyword and use LOAD DBID=, you can do a load of a full database. For rules that apply, see the information about loading a database on Loading a Database (see page 520).

The LOAD works by opening all areas in the database, acquiring KBYTES for each area, and adding each input record to the correct area as required. The order of the input records is therefore not important. 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)

,DDNAME=

Specifies the JCL DDname 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=.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to JCL specifications

Default Value:

(No default)

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)

,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 in z/VSE or the QSAM limit in z/OS.

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 DDNAME= 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)

This number multiplied by 1024 is the number of bytes requested through GETMAIN. 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.

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 MUF 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.

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 MUF and Single User modes. The database involved must be able to be opened for update in the MUF 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 MUF 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 MUF. With MULTUSE=YES:

- Area is first opened in MUF 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 MUF address space. Packages of index entries (about 32k bytes) are sent from utility to MUF.

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, the function executes with all index and data area data sets in the user address space and the CXX in MUF if running Simplify and MUF is enabled or else the CXX in the DBUTLTY address space.

For more information on area level control, see Area Level DBUTLTY Control.

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.

Important! 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=

Specifies the statistics display option to be passed to the sort program (see the SORT= parameter).

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

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. The system does not edit anything that you place inside of the quotes.

Valid Entries:

Any values that your installation's sort allows, as described in the previous paragraph.

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

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 Datadictionary.

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. The OPTION1= parameter is also highly recommended to obtain warning and critical messages if a problem occurs.

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 statements in a single step, place the largest sort value first.

If a number is supplied to the sort, 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.

CA-SORT Users (with SORT generated with the CHECK option): In the event that duplicate Master Key records are found and LOADDUPS=NO, CA Datacom/DB snaps the key value in error, and CA-SORT issues a message of the record counts being off and abends with a code of 16. The DB13001 message with a return code 10 is not issued.

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 available 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), 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)

,SORTDD=

Specifies an optional DD statement, provided to the SORT that redirects SYSPRINT-type output from the SYSPRINT DD (used by DBUTLTY) to this optional DD statement.

Valid Entries:

printed output directed to optional DD statement

Default Value:

printed output directed to SYSPRINT, intermixed with DBUTLTY output

,SORTDFLT=

(z/OS only) Specify SORTDFLT=YES when you want the DBUTLTY SORT= value to be overridden with what is expected to be a more accurate number.

SORTDFLT= can be set to YES or NO (the default). If SORTDFLT=YES, the SORT= value is required to be greater than zero.

Note: The SORTDFLT option has no equivalent in the DBFLSUB process.

The keyword, SORTDFLT= allows the DBUTLTY SORT=value, when specified as greater than zero (0) for a LOAD, to be overridden with an estimate expected to be more accurate. SORTDFLT= can only be used in conjunction with the SORT= keyword. When set as SORTDFLT=YES, the LOAD looks for record counts in the control records of the BACKUP input or the current CXX to use for the SORT= override. If the count in both places is zero (no records) for all tables to be loaded, the SORT= value is used and not overwritten. Verify that it is in the appropriate range to allow a successful LOAD.

The record count and the number of key definitions per table are multiplied for each table to be loaded and the products are summed as the estimate for record index entries. Based on the areas space management (DSOP) options, additional counts are added to verify a successful sort.

When using this DBUTLTY LOAD enhancement, consider the following:

- The number of records in a table is not always known.
- The number of records in a table after a MUF outage is likely to be inaccurate.
- A backup taken before Version 12.0 does not include information regarding the record count.
- The backup with Version 12.0 or later has the count as stored in the CXX at the time of the backup.
- Extract files never have a count, because they have no control records.
- For each table, the record count in the BACKUP control record is used, if known and not zero, otherwise, the current CXX record count is used. Either could be more accurate.

- The record count information from BACKUP input is taken from the control records that exist before the first data record, and any use of concatenated input is inappropriate to using the SORTDFLT=YES option, for example, if a database has two areas and they are each backed up separately. That is, if the two backups are concatenated into one base load, the control records at the start of the first input provide information about the first area, but the control records for the second input are not seen until the first area is complete, which is therefore too late to use to start the SORT. The pairing of base backups with base loads and area backups with areas loads do not have this issue.

Valid Entries:

NO or YES

Default Value:

NO

,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)

,UNIT=

Specifies the unit name for sort work areas. The parameter is always edited, but is only used when the SORT= parameter is specified as not zero.

Valid Entries:

1- to 8-character unit name

Default Value:

SYSDA

,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 Datadictionary 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

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 or 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.

```
//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
//BACKDEM  DD DSN=input                          Input data set
//*
//* NOTE: The following sort work allocations are not
//*       required if your sort package can dynamically
//*       allocate sort work areas.  Be sure to provide
//*       sufficient space for all keys to be sorted.
//*
//SORTWK01 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SORTWK02 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SORTWK03 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SYSIN    DD *                                  Command Input
        LOAD  AREA=DEM,DBID=1,FORMAT=BACKUP,CLUSTER=DENSE,
              DDNAME=BACKDEM,LOADDUPS=NO,OPTIMIZE=YES,SORT=500
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
LOAD   AREA=DEM,DBID=1,FORMAT=BACKUP,CLUSTER=DENSE,
         DDNAME=BACKDEM,LOADDUPS=NO,OPTIMIZE=YES,SORT=500

FUNCTION=LOAD
   AREA=DEM
   CLUSTER=DENSE
   DBID=1
   DDNAME=BACKDEM
   FORMAT=BACKUP
   LOADDUPS=NO
   OPTIMIZE=YES
   SORT=00000500
```

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.

```
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

This page of the report shows the following:
```

This page of the report shows the BACKUP FILE INFORMATION that 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. MVS users sometimes concatenate input files or use the JCL MOD option when producing a backup. 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, after it will, if necessary, occur the message **DB13012E INSUFFICIENT SPACE IN aaa AREA** or the message **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.

```
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
```

This page of the report shows the following:

INPUT FILE INFORMATION

This set of columns 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.

OUTPUT TABLE INFORMATION

This set of columns 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. For sample reports showing partitioning in OUTPUT TABLE INFORMATION, see Sample Report (Full Load with Partitioning) and Sample Report (Area Load with Partitioning).

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.

OUTPUT AREA INFORMATION

This set of columns displays 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.

Following is an example in which the AREA= keyword has not been specified, making this an example of a full database load. It is also an example of a full database load from a database with no partitioned tables in it (DBID 789 in the example) to a database with partitioned tables in it (DBID 997 in the example).

Assume that the INIT of the index, that must precede a full database LOAD, has already been done and was reported on pages one and two (the example therefore begins on page three).

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 LOAD   DBID=997,DDNAME=BACKUP,FORMAT=BACKUP,
SORT=100,LOADID=NO

FUNCTION=LOAD
   DBID=00997
   DDNAME=BACKUP
   FORMAT=BACKUP
   LOADID=NO
   SORT=000000100
```

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.

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA
JOB001     31,048  mm/dd/ccyy hh.mm.ss  PRODCXX     789  NAT   NO
INPUT FILE INFORMATION          OUTPUT TABLE INFORMATION
AREA  TABLE    ID     RECORDS     AREA  TABLE    ID     RECORDS     NOTES
                                  AK1   K11-K01 110           0
                                  AK2   K12-K01 111           0
                                  AK3   K13-K01 112           0
A01   C01     101          99     A11   C11-C01 102           9
                                  A12   C12-C01 103           0
                                  A13   C13-C01 104          90
A01   C02     102          99     A21   C21-C02 106          10
                                  A22   C22-C02 107          10
                                  A23   C23-C02 108          79
A01   F01     103          99                                    TABLE NOT LOADED
A01   F02     104          99                                    TABLE NOT LOADED
A01   F03     105           0                                    TABLE NOT LOADED
A01   RNA     106           0                                    TABLE NOT LOADED
A01   RNB     107           0                                    TABLE NOT LOADED
OUTPUT AREA INFORMATION
          TOTAL       USED       TOTAL       USED PCNT    AVG     BLOCKS     BLOCKS       TOTAL
AREA     TRACKS     TRACKS     BLOCKS     BLOCKS FULL  SLACK   EXPANDED COMPRESSED     RECORDS
AK1           8          1         96          2    2  4,094        N/A        N/A           0
AK2           8          1         96          2    2  4,094        N/A        N/A           0
AK3           8          1         96          2    2  4,094        N/A        N/A           0
A11           5          1         60          2    3  4,094        N/A        N/A           9
A12           5          1         60          2    3  4,094        N/A        N/A           0
A13           5          1         60          5    8     62        N/A        N/A          90
A21           5          1         60          3    5  3,454        N/A        N/A          10
A22           5          1         60          3    5  3,445        N/A        N/A          10
A23           5          1         60          4    6      2        N/A        N/A          79
```

This page of the report shows the following:

BACKUP FILE INFORMATION

For a description of the BACKUP FILE INFORMATION, see the explanation following Sample Report: LOAD Data Area.

INPUT FILE INFORMATION

For a description of the INPUT FILE INFORMATION, see the explanation following Sample Report: LOAD Data Area.

OUTPUT TABLE INFORMATION

For a description of the OUTPUT TABLE INFORMATION, see the explanation following Sample Report: LOAD Data Area.

Note: This sample OUTPUT TABLE INFORMATION shows partitioning.

OUTPUT AREA INFORMATION

For a description of the OUTPUT AREA INFORMATION, see the explanation following Sample Report: LOAD Data Area.

Following is an example of an area level load of a database with no partitioned tables in it (DBID 789 in the sample) into a database that has partitioned tables in it (DBID 997 in the sample).

AREA=A11 was specified, and the previous example (showing a full load of database 997) lists area A11 as containing only 9 records. This example therefore shows 9 records (rows) having been loaded and 90 skipped (LOADPTN=SKIP was chosen).

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 LOAD   DBID=997,DDNAME=BACKUP,FORMAT=BACKUP,
SORT=100,LOADID=NO,LOADPTN=SKIP,AREA=A11

FUNCTION=LOAD
   AREA=A11
   DBID=00997
   DDNAME=BACKUP
   FORMAT=BACKUP
   LOADID=NO
   LOADPTN=SKIP
   SORT=000000100
```

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.

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA
JOB002     31,048  mm/dd/ccyy hh.mm.ss  PRODCXX     789  NAT   NO
INPUT FILE INFORMATION          OUTPUT TABLE INFORMATION
AREA  TABLE    ID     RECORDS     AREA  TABLE    ID     RECORDS     NOTES
A01   C01     101          99     A11   C11-C01 102           9     SKIPPED
A01   C02     102          99                                       TABLE NOT LOADED
A01   F01     103          99                                       TABLE NOT LOADED
A01   F02     104          99                                       TABLE NOT LOADED
A01   F03     105           0                                       TABLE NOT LOADED
A01   RNA     106           0                                       TABLE NOT LOADED
A01   RNB     107           0                                       TABLE NOT LOADED
OUTPUT AREA INFORMATION
          TOTAL       USED      TOTAL       USED PCNT    AVG     BLOCKS     BLOCKS       TOTAL
AREA     TRACKS     TRACKS     BLOCKS     BLOCKS FULL  SLACK   EXPANDED COMPRESSED     RECORDS
A11           5          1         60          2    3  4,094        N/A        N/A           9         90 RECDS SKIPPED
```

This page of the report shows the following:

BACKUP FILE INFORMATION

For a description of the BACKUP FILE INFORMATION, see the explanation following Sample Report: LOAD Data Area.

INPUT FILE INFORMATION

For a description of the INPUT FILE INFORMATION, see the explanation following Sample Report: LOAD Data Area.

OUTPUT TABLE INFORMATION

For a description of the OUTPUT TABLE INFORMATION, see the explanation following Sample Report: LOAD Data Area.

Note: This sample OUTPUT TABLE INFORMATION shows partitioning.

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.

OUTPUT AREA INFORMATION

For a description of the OUTPUT AREA INFORMATION, see the explanation following Sample Report: LOAD Data Area.

SORTDFLT=YES Report Section for LOAD

If SORDTDFLT= YES, an additional section to the LOAD report is added after the optional BACKUP INFORMATION section, if present, and before the INPUT FILE INFORMATION section. The heading of the new section is SORTDFLT INFORMATION. After the heading, one line is printed for each table to be loaded.

The line after the SORTDFLT INFORMATION heading contains the following:

- Table name and ID
- Record count from the BACKUP control record for the table - The number is set as 0 if there is no input, extract input, or backup taken prior to Version 12.0. For a Version 12.0 backup, it is the count in the CXX at the time of the backup and might or might not be accurate.

  Note: This report section occurs at the time the first data row is read from the input. Only the control information available at this time is reported. The use of concatenated input with additional control information is subject to the INPUT FILE INFORMATION section but not the SORTDFLT INFORMATION section.
- Record count from the current CXX
- Count of key definitions for the table
- Estimated count of keys to sort using the backup count if the count is not zero, else the Directory count.

The next line starting with *DSOP is the count of index entries possible based on the data space options for the areas being loaded. If the estimated count of keys from the record input is all zeros then the *DSOP value is not added into the total and the override does not occur.

Following the *DSOP line is TOTAL and provides the sum of the key estimates from the SORTDFLT INFORMATION. It also provides information about what is overriding the SORT= keyword.

If the keyword FORMAT= is set to NONE, the sort value is set to one. If the keyword FORMAT= is not set to NONE, in the case where the record counts are all zero, CA Datacom/DB has no information to use for estimating the value of keys to be sorted. In this case, the SORT= value is not overridden. Estimate the values of the keyword SORT= with as much accuracy as possible.

The values are necessary for the sort to estimate the workspace needed and the technique to be used.

If the record counts are significantly incorrect but not zero, the sort uses the estimates as they exist. If the LOAD fails because of low sort count issues, change the JCL to provide a more accurate SORT= value then rerun the LOAD without using the SORTDFLT= keyword.

Example

For this example, database 2 was used. The first statement is a RETIX with KEYNAME=*SETR which recounts the records in the area; it was used only to ensure the counts were accurate in the example and is not intended for normal use.

```
RETIX  AREA=DD1,DBID=2,KEYNAME=*SETR
COMM  OPTION=CLOSE,DBID=2
REPORT AREA=CXX,DBID=2,TYPE=A
BACKUP DBID=2,SEQ=NAT,DDNAME=Y
INIT   DBID=2,AREA=IXX
LOAD DBID=2,FORMAT=BACKUP,DDNAME=Y,SORT=99999,SORTDFLT=YES,
OPTION1='(I)'
```

The resulting output is as follows:

```
BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA
DBDVL       2,812   8/30/2009 09.52.00  DBDVM0        2  NAT   NO
SORTDFLT INFORMATION
TABLE    ID       BACKUP    DIRECTORY  KEYS      ESTIMATE
  AGR    17          333          333     8         2,664
  ALS     9        1,857        1,857     3         5,571
  ARA     2           48           48     3           144
  ATZ    24            7            7     2            14
  BAS     1           23           23     5           115
  DVW    40           70           70     2           140
  ELM     7          580          580     5         2,900
  FCN    29            0            0     3             0
  FIL     3           86           86     2           172
  FLD     5       14,557       14,557    10       145,570
  HSD   123          196          196     1           196
  JOB    27            0            0     2             0
  KEY     6          702          702    12         8,424
  KWC     8           30           30     9           270
  LIB    56            0            0     2             0
  MEM    12            0            0     2             0
  MOD    23            0            0     2             0
  NOD    13            0            0     3             0
  PCV    20            0            0     2             0
  PER    25            2            2     2             4
  PGM    22            0            0     2             0
  PLN    15            0            0     3             0
  PNL    14            0            0     2             0
  PRC     4            0            0     3             0
  PRT    18            0            0     2             0
  REL    11        2,634        2,634     7        18,438
  RPT    26            0            0     2             0
  STM    16            0            0     3             0
  STP    28            0            0     2             0
  SYS    21           12           12     2            24
  TRG    19            0            0     3             0
  TXT    10          100          100     2           200
  *DSOP                                             38
  TOTAL (SORT ESTIMATE OVERRIDE)                 184,884   SORT=00000184884
```

```
INPUT FILE INFORMATION            OUTPUT TABLE INFORMATION
AREA  TABLE     ID      RECORDS     AREA  TABLE     ID      RECORDS     NOTES
DD1   AGR       17          333     DD1   AGR       17          333
DD1   ALS        9        1,857     DD1   ALS        9        1,857
DD1   ARA        2           48     DD1   ARA        2           48
DD1   ATZ       24            7     DD1   ATZ       24            7
DD1   BAS        1           23     DD1   BAS        1           23
DD1   DVW       40           70     DD1   DVW       40           70
DD1   ELM        7          580     DD1   ELM        7          580
DD1   FCN       29            0     DD1   FCN       29            0
DD1   FIL        3           86     DD1   FIL        3           86
DD1   FLD        5       14,557     DD1   FLD        5       14,557
DD1   HSD      123          196     DD1   HSD      123          196
DD1   JOB       27            0     DD1   JOB       27            0
DD1   KEY        6          702     DD1   KEY        6          702
DD1   KWC        8           30     DD1   KWC        8           30
DD1   LIB       56            0     DD1   LIB       56            0
DD1   MEM       12            0     DD1   MEM       12            0
DD1   MOD       23            0     DD1   MOD       23            0
DD1   NOD       13            0     DD1   NOD       13            0
DD1   PCV       20            0     DD1   PCV       20            0
DD1   PER       25            2     DD1   PER       25            2
DD1   PGM       22            0     DD1   PGM       22            0
DD1   PLN       15            0     DD1   PLN       15            0
DD1   PNL       14            0     DD1   PNL       14            0
DD1   PRC        4            0     DD1   PRC        4            0
DD1   PRT       18            0     DD1   PRT       18            0
DD1   REL       11        2,634     DD1   REL       11        2,634
DD1   RPT       26            0     DD1   RPT       26            0
DD1   STM       16            0     DD1   STM       16            0
DD1   STP       28            0     DD1   STP       28            0
DD1   SYS       21           12     DD1   SYS       21           12
DD1   TRG       19            0     DD1   TRG       19            0
DD1   TXT       10          100     DD1   TXT       10          100
OUTPUT AREA INFORMATION
          TOTAL       USED      TOTAL       USED PCNT    AVG     BLOCKS     BLOCKS       TOTAL
AREA     TRACKS     TRACKS     BLOCKS     BLOCKS FULL  SLACK   EXPANDED COMPRESSED     RECORDS
DD1         180        103      2,160      1,228   56    127        N/A        N/A      21,237
```

Information from SORT:

```
CASO000I  SORT    FILSZ=E00184884,FIELDS=(1,186,BI,A),DYNALLOC=(SYSDA,03)
CASO061I RCD IN 00182359,OUT 00000000
```

In this example, the count of keys sorted is less than the estimate because CA Datacom Datadictionary uses the rarely used NIL key feature to not be part of the index. NIL keys are defined to have a value of either all blanks (x'40') or all binary zeros (x'00').

Chapter 28: LOCK (Lock a Data Area)

The LOCK function allows you to back up a data area, specifically a compressed data area, while the MUF 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) for information.

A lock, when granted, remains in effect until an unlock is done or the MUF 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.

The LOCK function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

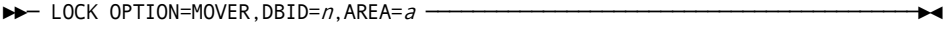
When to Use

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

The MUF must be active when you 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                Command Input
         LOCK      OPTION=MOVER,DBID=1,AREA=DEM
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    LOCK      OPTION=MOVER,DBID=1,AREA=DEM

                    FUNCTION=LOCK
                       AREA=DEM
                       DBID=1
                       OPTION=MOVER
```

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.

```
                *                                   REQUEST COMPLETE
```

This page of the report shows a message indicating that the LOCK completed successfully.

Chapter 29: MASSADD (Add Records to Table)

The MASSADD function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

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 MUF, 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 MUF, 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 Datadictionary with DBEDITS=Y.

If any field in the table is defined in CA Datacom Datadictionary 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 MUF

MASSADD can perform the add requests either through the MUF or as a Single-User task. You must run MASSADD through the MUF to add records to a table defined with integrity constraints.

We recommend that you run MASSADD with the MUF 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 MUF, 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, Simplify NO, or MUF down*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any MUF for read.
- The database may *not* have any table in *unloading* status.

Environmental **Requirements** *when MULTUSE=NO, Simplify YES, and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when MULTUSE=NO, Simplify NO, or MUF down*

- 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 **Controls** *when MULTUSE=NO, Simplify YES, and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions

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

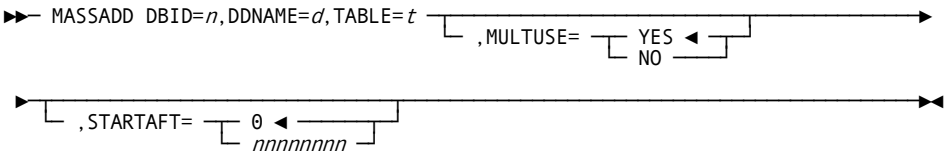
- The database may or may not be open for update in the connected MUF.
- The database may *not* be open for update in another MUF or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- 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 MUF for update.
- The utility does *not* set ACCESS area.
- The utility does *not* set MASSADD executing this MUF 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)

,DDNAME=

Specifies the JCL DDname of the input data. If the input source is a VSAM file, CA Datacom/DB processes it as an ordinary file.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to 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

,MULTUSE=

Specifies whether the add commands are to be directed to the MUF.

YES

MASSADD directs its input to the MUF. Direct commands to the MUF 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

,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 an active MUF 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.


//TELLER   DD DSN=input                              Input data set
//SYSIN    DD *                                      Command Input
         MASSADD   DBID=400,DDNAME=TELLER,TABLE=TEL
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
MASSADD DBID=400,DDNAME=TELLER,TABLE=TEL

FUNCTION=MASSADD
   DBID=400
   DDNAME=TELLER
   TABLE=TEL
```

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.

```
RECORD ******1000 SUCCESSFULLY ADDED
RECORD ******2001 SUCCESSFULLY ADDED
RECORD ******3000 SUCCESSFULLY ADDED

               ADDED

    TEL        3,400
```

This page of the report shows the following:

RECORD

A line is produced for each 1,000 records successfully added.

ADDED

Report indicates that 3,400 Records were added to the table.

Chapter 30: 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, the table being reorganized was unavailable while backups and reloads of the data, in Native Key sequence, were performed. The OLREORG function lets you reorganize the database table without taking it offline.

For information about support for the OLREORG function by the DBUTLTY STATUS console command, see DBUTLTY STATUS.

Note: The DBUTLTY STATUS command is only available when either the FIRSTKEY= and LASTKEY= keywords are not specified or no rows are found in the specified key range.

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 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 about 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. This statistical information is contained in the Area Daily Statistics (ADS) table in the history database. We therefore recommend that you enable ADS before using OLREORG.

Note: For information about the history database and ADS, see the *CA Datacom/DB Database and System Administration Guide*.

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 about the REFGROUP report, see REFGROUP Report.

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 MUF 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.

As OLREORG moves rows from block to block to improve order, it als frees space in blocks. You can rerun it and it will often find additional movements that would be beneficial.

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 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.

Using Empty Blocks in OLREORG

When the amount of free space is limited in used data blocks, the DBUTLTY function OLREORG allows the use of empty blocks in a reorganization.

Support for the OLREORG empty block feature includes the following categories:

- Implementation of a way to keep track of empty blocks
- New DBUTLTY behaviors
- Changes to the REPORT produced by OLREORG
- External limits

Keeping Track of Empty Blocks

CA Datacom/DB keeps track of the available space of a data area using a special key ID in each DBID index. Data space option 1, basic space reclamation, and 2, wraparound mode, do not keep track of which blocks are logically empty, that is, blocks that were used in the past but currently have all their rows deleted.

To enhance the efficiency of OLREORG, the following space options were added that keep track of empty blocks:

- Option 4 is basic space reclamation with an empty block index.
- Option 5 is wraparound mode with an empty block index.

If you plan to use OLREORG, then it is appropriate to migrate all DSOP option 1 areas to DSOP option 4 and all DSOP option 2 areas to DSOP option 5.

Changing the space option does not affect the loaded status of the data area and is effective immediately following its change in the CXX. Because the empty block index was not maintained in the past for basic and wraparound modes, we recommend that, when changing to a DSOP 4 or 5, you run a RETIX KEYNAME=*DATA to rebuild the space index, including the new empty block component of the space index.

Note: You are not required to run this RETIX.

DBUTLTY OLREORG Report Example Change

If the DSOP area is 4 or 5, the line is: *Empty blocks used.*

The line displays a count of the number of empty blocks to which rows were moved during the OLREORG. The following is a sample of this report:

```
Rows processed -                      5,000
Rows moved -                          2,512
Blocks freed in reference group -     2,510
Exclusive control conflicts -             0
Empty blocks used -                       5
```

If the DSOP area is 0, 1, or 2, the line is: *No empty blocks used due to DSOP.*

Note: A DSOP of 3 is not supported by OLREORG.

The following is a sample of this report:

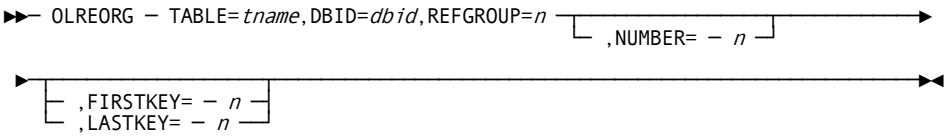
```
Rows processed -                      5,000
Rows moved -                            284
Blocks freed in reference group -       284
Exclusive control conflicts -             0
No empty blocks used due to DSOP
```

External Limits

If there are no more blocks in the empty block index, OLREORG uses the next available overflow (NAO) block. OLREORG never causes a dynamic extend.

How to Use

The MUF must be active when you 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 to be 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
       OLREORG TABLE=ALS,DBID=2,REFGROUP=50
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

For sample REFGROUP reports, see Sample Report: REFGROUP Summary. An example of the summary report generated by executing the OLREORG function against a table containing 3,000 rows follows:

```
Rows processed -                3,000

Rows moved -                      384

Blocks freed in reference group -   129

Exclusive control conflicts -       0
```

The report displays the following information:

Rows processed

Rows processed is the total rows in the table, or if using FIRSTKEY or LASTKEY, the total rows within the specified key range.

Rows moved

Rows moved is the total rows moved from less heavily populated blocks within a reference group to more heavily populated blocks.

Blocks freed in reference group

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.

Exclusive control conflicts

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 31: PREINIT (Pre-Format Index and Data Areas)

The PREINIT function pre-formats (pre-initializes) a planned non-existing index area or data area. See PREINIT index area (see page 577) and PREINIT data area. (see page 585)

PREINIT promotes 24X7 operations. When you use PREINIT to pre-format a new index area, to contain a key to be cataloged in the 24x7 way using a CA Datacom Datadictionary 1000 APPLYCXX batch transaction, the database can be left open for all normal user applications while the key is built. Without PREINIT, cataloging a key into an index area requires that the area already exists and is in use by another key. Also, use PREINIT to pre-format a new data area to contain a table to be added using SQL DDL statements to CREATE a new table. If you do a PREINIT of an area that is later decided to have been done in error or is no longer needed, you can clean up the PREINIT list. To delete an entry on the PREINIT list, use the following:

```
CXXMAINT OPTION=DPREINIT,DBID=n,AREA=aaa
```

The PREINIT list is reported using the DBUTLTY function REPORT with AREA=CXX,TYPE=P. It reports on all items in the PREINIT list. If the list is empty, it has nothing to report. The first page of the full CXX report (not type A) and the same report requested as TYPE=K provides PREINIT AREAS – NO (or YES) to alert users whether any PREINIT areas exist.

PREINIT index area updates with control information the first block in the pre-formatted index area. The first block is subsequently updated when maintenance is performed. The default block size is 4096 for all devices, except for 3380 devices that have a block size of 3860. The block size can be set using the optional keyword BLKSIZE=.

You can also use PREINIT to pre-format and null load a data area that does not already exist. PREINIT data area makes it possible for SQL to be used to create a table in a new area and, optionally, to create a table in a new database. The table is then immediately loaded and available, and the SQL create process can successfully complete.

Note: Without PREINIT, SQL can only create a table if that table is created as loaded into an area that already has another table in it, and for which the area has already been loaded.

Summary of Requirements

- You use the PREINIT function for an index area or data area out of normal DBUTLTY context. For example, the PREINIT does not validate that the database does or does not exist. Wheher a database exists or not is not a concern to the operation of the PREINIT function itself. Neither does PREINIT validate whether the area name exists. PREINIT can be used for many cases to allow complete or nearly complete 24x7 processing. The data set used with PREINIT is usually new, but it is possible and acceptable to use PREINIT with an existing data set. If you use PREINIT with an existing data set, the data set should neither be open nor a closed area containing data that is not intended to be overlaid.
- Using PREINIT is expected to be a rare event, with a PREINIT list item lasting a few minutes at most. PREINIT therefore requires minimal reporting.
- The PREINIT list exists at the CXX level and is reported in a REPORT function requested with AREA=CXX,TYPE=P.

- The PREINIT function requires the basic information needed for a successful INIT, including the area name and database ID (DBID). For PREINIT data area, the block size is required, but for PREINIT index area, the block size is optional.
- PREINIT does not open the database. PREINIT only opens the CXX.
- The standard DDNAME used in an INIT is used during a PREINIT, but the database is not opened. Also, in a PREINIT, if the area exists and has a data set name, it is not known, meaning that the DD statement is required. We recommended using PREINIT with DISP=NEW or DISP=OLD. You can use PREINIT with DISP=SHR, too, except when DISP=SHR is specified, in which case CA Datacom upgrades the DISP=SHR to a DISP=OLD specification. DISP=OLD provides protection against the data set being currently in use. The PREINIT function is for new areas and not for databases with existing areas.
- PREINIT formats the data set as the type of area needed. PREINIT formats the entire area for a data area to match the combination of a data INIT and null LOAD.
- PREINIT updates the CXX with a new structure of a list of PREINIT data sets that includes the area type of either index or data, the area name, the block size, the number of tracks, and the data set name. If the area and database match an existing PREINIT entry, the previous entry is replaced by the current entry.
- An item on the PREINIT list can be deleted using CXXMAINT OPTION=DPREINIT,DBID=n,AREA=aaa.
- An item on the list does not mean it currently exists as a data set, because users can delete or rename data sets outside of CA Datacom knowledge.

During a CA Datacom Datadictionary catalog, the following special actions occur:

- If the database is new, the list is searched for the IXX area in that database. If found, the entry is taken and the information saved as the current IXX, and the index is set as loaded.
- If the index area (not IXX) is new, the entry list is searched and, if found, the entry is taken.
- If the data area is new or has no Data Set Name (DSN), the list is searched and, if found, the entry is taken.

On the first normal open of a data set that was built using PREINIT, the information that would normally be in a control block is added. An edit ensures that the data set that was opened has been subjected to a PREINIT and has never been used since initialization.

The PREINIT list is intended to be short-lived. It is not part of a CXX backup function and is therefore not available as the subject of a CXX load.

When to Use PREINIT index area

Use the PREINIT function when you want to pre-format a new index area.

Successful Execution Requirements and Controls

Environmental **Requirements** *when Simplify NO or MUF down*

None.

Environmental **Requirements** *when Simplify YES and MUF enabled*

None.

Environmental **Controls** *when Simplify NO or MUF enabled*

None

Environmental **Controls** *when Simplify YES and MUF down*

None

Multivolume INITs

The data area can exist as one or more extents on one or more 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 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.

If the keyword VOLUMES= is specified with a number value that is less than the number of volumes provided, the INIT function only initializes the number specified in the VOLUMES= keyword. For example, for a DD statement with VOL=SER=(VOL001,VOL002,VOL003), without the VOLUMES= keyword DBUTLTY initializes 1 to 16 extents on VOL001, 1 to 16 extents on VOL002, and 1 (or more) extents on VOL003. But with the VOLUMES= keyword set as VOLUMES=2, DBUTLTY initializes 1 to 16 extents on VOL001 and one (or more) extents on VOL002. Unused extent possibilities on VOL002 and VOL003 are available for dynamic extend.

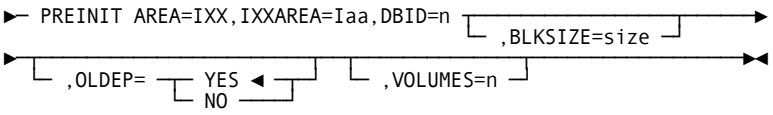
Controlling the Index Block Size

The block size of each database Index Area (IXX) and index partition area (Inn) is determined during IXX (or Inn) initialization. When you initialize an index area, 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 index area. Pick an optimal size for your device type, in a manner similar to how the data area block size is selected.

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. For more information about the *ixxbfrsize* parameter of the SYSPOOL MUF startup option, see *CA Datacom/DB Database and System Administration Guide*.

How to Use

To pre-format an index area, execute DBUTLTY using the following command format:



Command

PREINIT

Invokes the function to pre-format index areas for CA Datacom/DB use.

Required Keywords

AREA=IXX

Identifies the area to be pre-formatted as an index.

Valid Entries:

IXX

Default Value:

(No default)

,IXXAREA=I*aa*

Specify a specific index area to PREINIT using IXXAREA=I*aa* where I*aa* is an uppercase letter i (I) followed by two variable characters, *aa,* that can be XX or 00 through 99. Specify IXXAREA=IXX for the main index IXX. To specify the PREINIT of a separate index area to contain some of the key IDs within the database, use IXXAREA=I*aa* as previously described.

As in the syntax diagram previously shown, when pre-initializing an index area, use AREA=IXX in combination with any valid value for IXXAREA= I*aa*.

Valid Entries:

IXX or, I*aa*, where *aa* can be 00 through 99 as previously described

Default Value:

(No default)

,DBID=*n*

Identifies the database.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

Optional Keywords

,BLKSIZE=*size*

The BLKSIZE= 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 required keywords in How to Use.

We do not recommend changing the block size from the default. For more information, see Controlling the Index Block Size.

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. For more information about the *ixxbfrsize* parameter of the SYSPOOL MUF startup option, see 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. If you specify OLDEP=NO, you ensure that DBNTRY is used. Using DBENTRY 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.

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

,VOLUMES=*n*

(*z/OS only*)The *n* in VOLUMES=*n* is the maximum number of volumes to initialize. Without the VOLUMES= option, the number of volumes having specific VOLSER information is initialized. The VOLUMES= option allows more volumes to be catalogued as part of the data set and available to an EXTEND or dynamic EXTEND, while being effectively held in reserve without a specific allocation.

For example, you could specify VOLUMES=3 when using a PREINIT to initialize an area that needs 3 volumes, intending to only initialize the space on 3 volumes. But the JCL could specify 6 specific volumes, for example, VOL=SER=(VOL001,VOL002,VOL003,VOL004,VOL005,VOL006), so that volumes 4/5/6 are not allocated or used during the INIT but are available to a dynamic extend. In this case, the extra volumes would not need to be removed before another INIT.

Valid Entries:

the number *n* of volumes to be catalogued

Default Value:

(No Default)

Examples

For an example CXX PREINIT Report, see Type P - PREINIT Report (see page 683).

Example 1

Use the PREINIT function if you want to add a new key to a new index area (identified as index area I10 in this example) to an open and loaded database in a 24x7 environment.

The catalog of a new key requires an index area that is initialized and loaded. To meet the 24x7 requirements, you can use the PREINIT function to create a new index area data set and initialize it before there is a need to add the new key to a new index area.

Specify the keywords DBID=, AREA=IXX, and IXXAREA=. Optional for the PREINIT for an index are OLDEP=, BLKSIZE=, and VOLUMES=. The optional keywords work the same with PRENIT as with INIT.

DBUTLTY opens the data set out of all normal context (such as not opening the database). The area is initialized with the provided values. The data set name is saved in the CXX in the PREINIT list.

After completion of the PREINIT, the key catalog can be done. When a key catalog discovers that the index area to be used contains no other key and therefore does not exist, the PREINIT list is searched to discover the initialized area. The key catalog locks in the data set name to the new index and deletes the PREINIT list item. When the index is opened during the addition of the key, the current data set is validated. If the data set is not found or the wrong database or area is found, it is rejected and the key catalog is aborted. The failed addition of the key needs to be deleted and the process attempted from scratch. A database being opened must find that all index areas are properly defined and initialized. If the PREINIT process is not followed correctly, the database is not opened until the new key to be added is deleted from CA Datacom Datadictionary and the database is cataloged. Otherwise, a new index area must be properly subjected to an INIT function.

The PREINIT list is only accessed during a catalog and an SQL CREATE for an area that is not known to the current database.

If requirements are met, the key can be added in a 24x7 environment in parallel with the database being opened for user processing.

Example 2

You can use the PREINIT function in concert with SQL DDL statements to add a new database and a new table and have the new database and table be instantly loaded and available. You can also add a new table into a new area in an existing database.

In this example, use PREINIT with the required keywords DBID=, AREA=IXX, and IXXAREA=IXX. Other optional keywords available when AREA=IXX is used are OLDEP=, BLKSIZE=, and VOLUMES=. Next use PREINIT to initialize each data area in the database. For a data area, the required keywords are DBID=, AREA=, and BLKSIZE=. Other optional keywords available are VOLUMES=.

The standard DDNAME is used during a PREINIT, but the database is not opened and if the area exists and has a data set name, it is not known. The DD statement is therefore required.

Example JCL

The following shows the command to pre-format an index area with a DBID of 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.

```
 //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
//I01010 DD DSN=data.set.name,DISP=(,CATLG),other keywords
           PREINIT     AREA=IXX,DBID=10,IXXAREA=I01
 /*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Using PREINIT is expected to be a rare event, with a PREINIT list item lasting a few minutes at most. PREINIT therefore requires minimal reporting. The PREINIT list exists at the CXX level and is reported in a REPORT function requested with AREA=CXX,TYPE=P.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
PREINIT AREA=IXX,IXXAREA=I01,DBID=10

FUNCTION=PREINIT
   AREA=IXX
   IXXAREA=I01
   DBID=10
```

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.

```
I01 - STATUS INFORMATION

DEVICE TYPE -    3380
BLOCK LENGTH -  3,860
BLOCKS/TRACK -     11

BLOCKS IN USE -         2
BLOCKS UNUSED -       108
TOTAL BLOCKS -        110
TOTAL TRACKS -         10
PERCENT FULL -          2
```

This page of the report shows status information about the pre-formatted index area at successful completion of pre-initialization.

When to Use PREINIT data area

Use PREINIT data area when you want to pre-format and null load a data area that does not already exist.

PREINIT data area makes it possible for SQL to be used to create a table in a new area using the SQL statement CREATE AREA and, optionally, to create a table in a new database using the SQL statement CREATE TABLE. The table is then immediately loaded and available. The SQL create process can subsequently successfully complete. Without PREINIT, SQL can only create a table if that table is created as loaded into an area that already has another table in it, and for which the area has already been loaded.

Multivolume INITs

The data area can exist as one or more extents on one or more 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, Simplify NO, or MUF down*

None.

Environmental **Requirements** *when MULTUSE=NO Simplify YES and MUF enabled*

None.

Environmental **Controls** *when Simplify NO or MUF enabled*

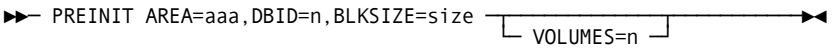
None.

Environmental **Controls** *when Simplify YES and MUF down*

None.

How to Use PREINIT data area

To pre-format a data area, execute DBUTLTY using the following command format:



Command

PREINIT

Invokes the function to pre-format data areas for CA Datacom/DB use.

Required Keywords

AREA=*aaa*

You can PREINIT a data area with PREINIT AREA=*aaa* where *aaa* Identifies the data area to be pre-formatted. To PREINIT a data area, the *aaa* cannot be IXX.

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)

,BLKSIZE=*size*

BLKSIZE is the size of the data area blocks within the data area. The size is based upon the tables that are to be stored in the area. For PREINIT data area, specify the *size* in BLKSIZE=*size* to match the block size that you intend to provide in the CA Datacom Datadictionary definition for the data area.

Valid Entries:

512 - 32767

a valid block size

Default Value:

(No default)

Optional Keywords

,VOLUMES=*n*

The *n* in VOLUMES=*n* is the maximum number of volumes to initialize. Without the VOLUMES= option, the number of volumes having specific VOLSER information is initialized. The VOLUMES= option allows more volumes to be catalogued as part of the data set and available to an EXTEND or dynamic EXTEND, while being effectively held in reserve without a specific allocation.

For example, you could specify VOLUMES=3 when using a PREINIT to initialize an area that needs 3 volumes, intending to only initialize the space on 3 volumes. But the JCL could specify 6 specific volumes, for example, VOL=SER=(VOL001,VOL002,VOL003,VOL004,VOL005,VOL006), so that volumes 4/5/6 are not allocated or used during the INIT but are available to a dynamic extend. In this case, the extra volumes would not need to be removed before another INIT.

Valid Entries:

the number *n* of volumes to be catalogued

Default Value:

(No Default)

Examples

Example 1

Use the PREINIT function with required keywords DBID=, AREA=, and BLKSIZE= (plus, there is one optional keyword, VOLUMES=) if you want to use SQL DDL statements to add a new table in a new data area to a loaded and open database, and you want to have the new table be instantly loaded and available. Without PREINIT, a new table can only be set as loaded if it is going into an existing area with another table set as loaded. To meet the requirements, the data area needs to be initialized and loaded before the new table and area can be cataloged, and PREINIT does that for you.

The standard DDNAME is used during a PREINIT, but the database is not opened, and if the area exists and has a data set name, it is not known, meaning that the DD statement is required. We recommended using PREINIT with DISP=NEW or DISP=OLD, but DISP=SHR is allowed if a DISP=SHR is not specified. If DISP=SHR is specified, CA Datacom upgrades it to DISP=OLD to provide protection ensuring that the data set is not currently in use. The PREINIT function is meant to be used for new areas, not for databases with existing areas.

DBUTLTY opens the data set out of all normal context (such as not opening the database). The area is initialized with provided values and is set to be loaded. The data set name is saved in the CXX in the PREINIT list.

After completion of the PREINIT, the table can be created by SQL where the area is found to be not loaded but the PREINIT list entry is found to meet the requirement. The list entry is used for the area and deleted from the PREINIT list. When opened, the area must be found and initialized and loaded with the values required by the SQL DDL CREATE statement (or the prior definition, done in CA Datacom Datadictionary.

Example 2

See the last of the second example in PREINIT index area examples (see page 583).

Example JCL

The following shows the command to pre-format 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.

```
//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
//dem001 DD DSN=data.set.name,DISP=(,CATLG),other-keywords
//SYSIN    DD *                                  Command Input
          PREINIT     AREA=DEM,DBID=1,BLKSIZE=4096
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Using PREINIT is expected to be a rare event, with a PREINIT list item lasting a few minutes at most. PREINIT therefore requires minimal reporting. The PREINIT list exists at the CXX level and is reported in a REPORT function requested with AREA=CXX,TYPE=P.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
PREINIT AREA=DEM,DBID=1,BLKSIZE=4096

FUNCTION=PREINIT
   AREA=DEM
   DBID=00001
   BLKSIZE=04096
```

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.

```
DEM - STATUS INFORMATION

DEVICE TYPE -    3380
BLOCK LENGTH -  1,024
BLOCKS/TRACK -     31

BLOCKS IN USE -         1
BLOCKS UNUSED -       154
TOTAL BLOCKS -        155
TOTAL TRACKS -          5
PERCENT FULL -          1
```

This page of the report shows messages indicating the status of the data area at successful completion of the initialization.

Chapter 32: 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 MUF 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 statements, it is therefore your responsibility to ensure the existence of the Directory (CXX) definitions 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.

Note: The RECOVERY function supports tapes from the GDG (Generation Data Group).

Use of URI

We recommend the Unique Row Identifier (URI) record ID format for all environments. Using the URI format reduces manual intervention during recovery. You can convert to the URI format using a data area LOAD (see LOAD (Build a Data Area) (see page 519)).

RANGEDT= Keyword

The RANGEDT= keyword gives you a clean and complete method of restricting a forward or backward recovery effort to a single limited time period.

For more information about the RANGEDT= keyword, see Optional Keywords following Forward Recovery **(see page 603).**

FORCE= Keyword

The FORCE= keyword in forward or backward recovery gives you a way to specify whether the recovery utility allows input without the first record being a CYCLE START.

For more information on the FORCE= keyword, see Optional Keywords following Forward Recovery **(see page 603).**

For backward recovery, note that the MISMATCH= keyword is required.

MULTUSE= Keyword

In support of area level DBUTLTY control (see Area Level DBUTLTY Control), using MULTUSE=YES in the RECOVERY OPTION=FORWARD function 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 more information on the MULTUSE= keyword, see Optional Keywords following Forward Recovery **(see page 603).**

If MULTUSE=YES has not been specified, no other task can be processing the database when you are running forward 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 Optional Keywords following Forward Recovery **(see page 603).**

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 MUF 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 MUF, 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 (see RXXFIX (Recover End-Of-File)).

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) return 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. (See BACKUP (Create Backups) of this manual and continuous processing in the *CA Datacom/DB Database and System Administration Guide* for further details.)

Processing Multiple Recovery Files

Multiple recovery files can be processed in a single execution of the utility. Specify the files using normal JCL. The files concatenate in the order in which they are created.

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.

MUF Connection

This function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

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)). 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 MUF 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

The MUF must be active when you 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 MUF.
- The database may *not* be open for update in another MUF 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 MUF for update.
- The utility does *not* set ACCESS area.
- The utility does *not* set RECOVERY executing this MUF this database or area.

How to Use Forward Recovery

Forward recovery can be run in an active MUF mode for one area or in a Single User mode while the MUF is up, providing that the database is not being used. To use the DBUTLTY RECOVERY function for forward recovery not by area, 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 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 46).

Forward Recovery Successful Execution Requirements/Controls

Environmental **Requirements** *when MULTUSE=NO, Simplify NO, or MUF down*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any MUF for read.
- The database may *not* have any table in *unloading* status.

Environmental **Requirements** *when MULTUSE=NO, Simplify YES, and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when MULTUSE=NO,Simplify NO, or MUF down*

- 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 **Controls** *when MULTUSE=NO Simplify YES and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

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

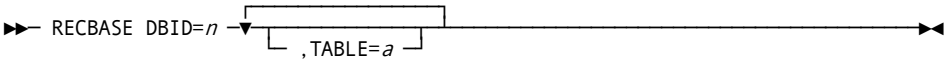
- The database may or may not be open for update in the connected MUF.
- The database may *not* be open for update in another MUF or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set 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 MUF if more than one is executing.

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

- The database will be opened by MUF for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets RECOVERY executing this MUF 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:

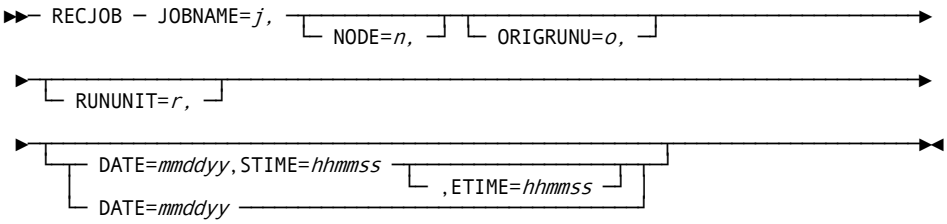
DATACOM-NAME of the table in the database specified

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.

NODE=

(For CA Datacom STAR users only.) Specifies the node at the originating site.

Valid Entries:

8-byte node designator

Default Value:

(No default)

ORIGRUNU=

(For CA Datacom STAR users only.) Specifies the run unit at the originating site.

Valid Entries:

5-character run unit

Default Value:

(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)

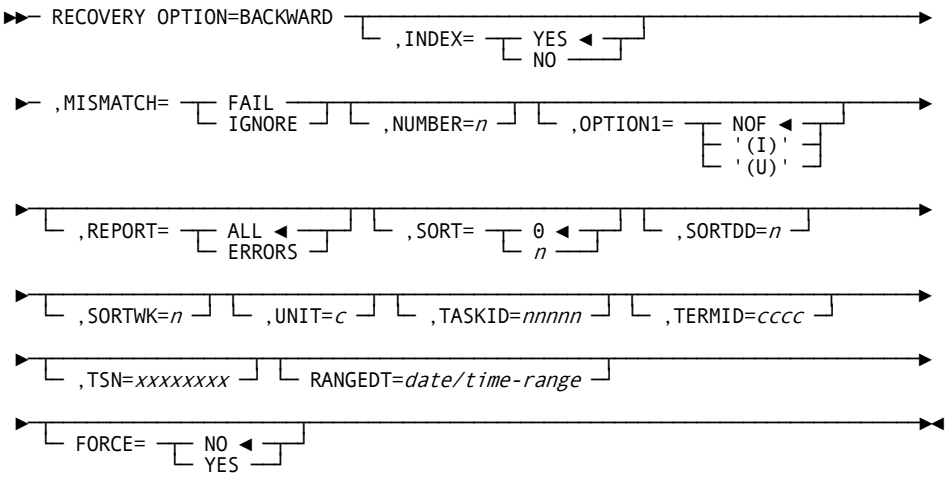
Default Value:

(No default)

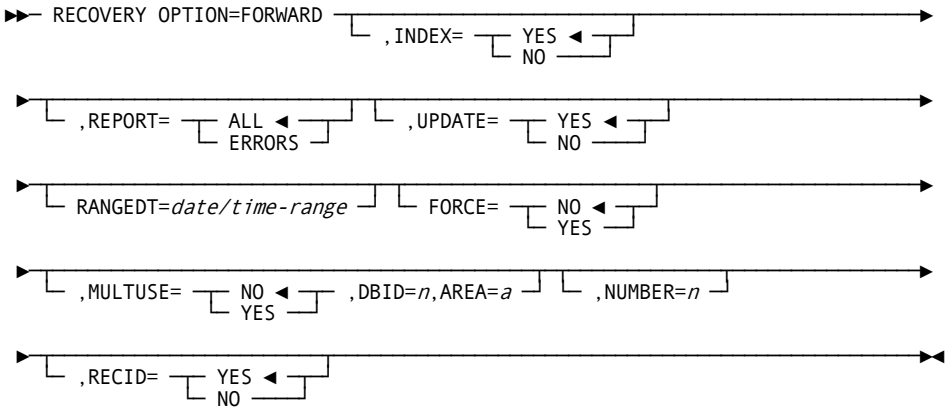
RECOVERY Command 3 (RECOVERY)

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.

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.

Valid Entries:

DATACOM-ID of the database

Default Value:

(No default)

For forward recovery only, 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 Recovery 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 utility 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

Specifies the number of input recovery files for forward recovery. Specify as 1 for backward recovery or let default to 1.

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 MUF. 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 MUF. 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.

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 or PXXML 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 CA 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=

Specifies the statistics display option to be passed to the sort program (see the SORT= parameter).

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. The OPTION1= 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 or '(I)' or '(U)'

Default Value:

NOF

Specifies the options for the associated sort/merge program application.

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. The system does not edit anything that you place inside of the quotes.

Valid Entries:

Any values your installation's sort allows are valid. 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 undesirable 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

SORTDD=

Specifies an optional DD statement, provided to the SORT that redirects SYSPRINT-type output from the SYSPRINT DD (used by DBUTLTY) to this optional DD statement.

Valid Entries:

Printed output directed to optional DD statement

Default Value:

Pinted output directed to SYSPRINT, intermixed with DBUTLTY output

,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

,UNIT=

Specifies the unit name for sort work areas.

Valid Entries:

1- to 8-character unit name

Default Value:

SYSDA

,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 MUF 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

For backward recovery when using SORT= see following Example 1. For forward recovery, see Example 2).

Note: For forward recovery the DDname must be specified as RXX.

Example JCL 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.


//RXX      DD DSN=rxx.data.set,DISP=SHR          Recovery data set
//SYSIN    DD *                                   Command Input
         RECBASE   DBID=1
         RECJOB    JOBNAME=PAYCHECK
         RECOVERY  OPTION=BACKWARD,SORT=1000,MISMATCH=FAIL
/*
```

The above example assumes that the job, ARMAINT, was running for longer than the two-hour period of time specified in the input parameters. The date and time keywords are not required to recover an entire job.

Example JCL 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.

```
//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
//RXX      DD DSN=rxx.data.set,DISP=SHR          Recovery data set
//SYSIN    DD *                                  Command Input
         RECBASE   DBID=1
         RECOVERY  OPTION=FORWARD
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
RECBASE DBID=1

FUNCTION=RECBASE
   DBID=00001
```

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.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
  RECOVERY OPTION=FORWARD

FUNCTION=RECOVERY
   OPTION=FORWARD
```

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.

```
                                    DATACOM/DB RECOVERY


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
```

This page of the report shows the following:

JOB NAME

The name of the job being recovered.

R-UNIT

The run unit of the job.

DATE

The date the record was logged.

TIME

The time the record was logged.

REQUEST

The contents of the request area.

BASE

The database ID.

BLOCK

The block number in hex.

URI

The Unique Row Identifier (URI) number in hex. For non-URI areas, the logical record number within the block in hex.

AREA

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.

R

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.

M

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.

START OF RECORD

The first 20 bytes of the user work area.

ERROR MESSAGES

The error messages produced during recovery.

```
                          DATACOM/DB RECOVERY JOBS NOT FOUND ON RECOVERY FILE


JOB NAME     DATE       START TIME     STOP TIME  RUN UNIT
```

This page of the report contains 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.

JOB NAME

The job name of the job for which recovery was requested.

DATE

The date of the job to be recovered.

START TIME

The start time for the recovery.

STOP TIME

The ending time for the recovery.

RUN UNIT

The run unit assigned to the job to be recovered.

```
     ERROR SUMMARY

TOTAL ERRORS ENCOUNTERED-----0

FIRST 01 PAGE(S) CONTAINING ERROR(S)-----4

LAST ERROR ON PAGE      -----4
```

This page is produced for all jobs and shows the following:

ERROR SUMMARY

A summary of the errors encountered.

Chapter 33: REMOVE (Remove Key ID)

The REMOVE function processes the Index Area for the specified database and deletes the entries for the specified key ID.

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 with MINIMAL=YES functions.
- Run the REMOVE and LOAD functions.
- Run the INIT and RETIX functions.
- Run the INIT and LOAD functions.

How to Use

The MUF must be active when you 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).

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.

```
//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
         REMOVE    AREA=IXX,DBID=1,KEYID=1
 /*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REMOVE AREA=IXX,DBID=1,KEYID=1

FUNCTION=REMOVE
   AREA=IXX
   DBID=1
   KEYID=1
```

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.

```
IXX - STATUS INFORMATION
DEVICE TYPE -    3380
BLOCK LENGTH -  3,860
BLOCKS/TRACK -     11

BLOCKS IN USE -
BLOCKS UNUSED -        55
TOTAL BLOCKS -         66
TOTAL TRACKS -          6
PERCENT FULL -         17
```

This page of the report indicates the status of the Index upon successful completion.

Chapter 34: REORG (Parallel BACKUP/LOAD)

The parallel REORG function of the CA Datacom/DB Utility (DBUTLTY) provides a way for you to quickly reorganize a large data area into *nearly 100 percent* Native Key sequence. The REORG function executes much faster than other reorganization options.

More Information:

- For information about DBUTLTY STATUS console command support for this function, see DBUTLTY STATUS.
- For information about the impact of table partitioning on the REORG function, see The REORG Utility (see page 46).

When to Use

REORG is best used with large areas that contain large amounts of data. We do not recommend using REORG to process small areas. For small areas, use the normal BACKUP and LOAD functions of the CA Datacom/DB Utility (DBUTLTY). Areas of medium size can best be reorganized by using the BACKUPONLY option of REORG to perform a fast parallel backup, followed by the resulting 1-25 files used as concatenated into a single LOAD function.

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

Three sets of subtasks are associated with the REORG function:

- Parallel backing up of an area in Native Key sequence
- Parallel loading of that data
- Parallel updating of the index

  Note: The REORG function updates a current index to reflect the moving of data into Native Key sequence order, but REORG does not reorganize an index, that is to say, REORG *does not build a new index.* REORG therefore cannot be successfully executed after the INIT of an index.

Instead of performing all three sets of subtasks with REORG, you can use the function to perform a parallel backup only, if desired, or you can use REORG to perform only a parallel loading of the data with index update. A *reporting only* option is also available, if you use REORG OPTION2=BACKUPONLY and either DDNAME=* or DDNAME**. For information about DDNAME=, 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, and 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 MUF. 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 is ignored, because the single user process has already opened the database for backup.

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. If OPTION2=BACKUPONLY was specified, the utility function is complete, and the data sets are available for use. If OPTION2= was not specified as BACKUPONLY, parallel data load starts. If this load fails, it must be restarted with the sequential files that were just produced, but with OPTION2= changed to LOADONLY. Temporary data sets should not be used unless, to allow for possible errors during the data load or index update phases, another form of backup was taken prior to the start of the REORG.

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, then input that output to the REORG OPTION2=LOADONLY function.

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.

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). The count is usually accurate.

DIAGOPTION 9,16,ON/OFF

Be aware of the following if, in r10, you had job streams specifying ACCESS STATUS=OFF in front of a REORG.

Controls exist to either prevent all access to a database (ACCESS STATUS=OFF) or allow all DBUTLTY functions that run in MUF (ACCESS STATUS=UTLTY). However, in r10 the backup portion of REORG ignored the access setting and executed where in later releases it is appropriately blocked from execution. DIAGOPTION 9,16,ON allows sites with the old (defective) JCL streams to continue to execute with disregard to the ACCESS OFF setting.

If this DIAGOPTION is used, then we recommend that you correct all necessary job streams to replace occurrences of ACCESS STATUS=OFF with ACCESS STATUS=UTLTY and remove the DIAGOPTION.

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 MUF 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.
- 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.
- With MULTUSE=YES, the DBUTLTY must execute on the same System on which the MUF is executing.

  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 BACKUPLOAD Execution Requirements/Controls

Environmental **Requirements** *when MULTUSE=YES/NO*

- The database may or may not be open for update in the connected MUF.
- The database may *not* be open for update in another MUF 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 will not be allowed against a Data Sharing MUF if more than one is executing.

Environmental **Controls** *when MULTUSE=YES/NO*

- The database will be opened by MUF 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 MUF this database or area.

LOADONLY or LOAD phase of BACKUPLOAD Execution Requirements/Controls

Environmental **Requirements** *when MULTUSE=NO, SIMPLIFY NO, or MUF down*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any MUF for read.
- The database may *not* have any table in *unloading* status.

Environmental **Requirements** *when MULTUSE=NO, Simplify YES and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when MULTUSE=NO, SIMPLIFY NO, or MUF down*

- 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 **Controls** *when MULTUSE=NO, Simplify YES and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

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

- The database may or may not be open for update in the connected MUF
- The database may *not* be open for update in another MUF 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 MUF if more than one is executing.

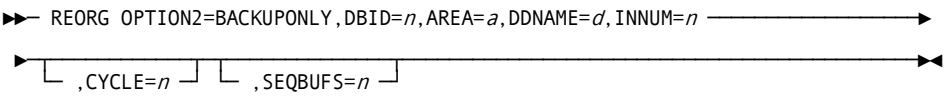
Environmental **Controls** *when MULTUSE=YES*

- The database will be opened by MUF for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets REORG-LD executing this MUF this database or area.

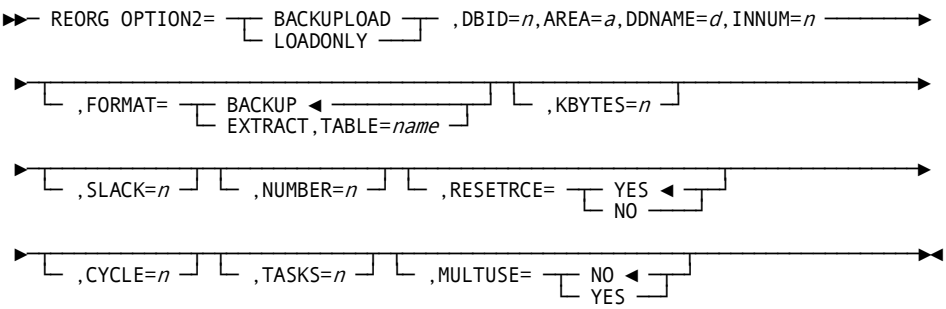
How to Use

The MUF must be active when you execute this command. Execute the REORG function using the following command format.

BACKUPONLY REORG



BACKUPLOAD/LOADONLY REORG



Note: FORMAT=EXTRACT is only valid when OPTION2=LOADONLY.

Command

REORG

Invokes the REORG function.

Required Keywords

OPTION2=

Specifies the type of REORG.

Valid Entries:

BACKUPLOAD, BACKUPONLY, LOADONLY

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 and/or loaded. up and/or loaded.

Valid Entries:

DATACOM-NAME of the area in the database specified in the DBID= parameter

Default Value:

(No default)

,DDNAME=

Specifies the JCL DDname to be used for the data input.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

A *reporting only* option is available, if you use REORG OPTION2=BACKUPONLY and either DDNAME=* or DDNAME** as described in the following:

DDNAME=*

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 or EXTRACT functions.

DDNAME=**

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 DDNAME.

Valid Entries:

Any name valid to JCL specifications (1-6 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 and/or load data.

Note: If you are partitioning by some portion of Native Key, running a REORG DDNAME=*,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

,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 MUF 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

,MULTUSE=

(For area level DBUTLTY control only.) MULTUSE=YES specifies area level DBUTLTY control that allows a single area in a database to be reorganized with no disabling of the other areas in the database. When MULTUSE=YES:

- The data spaces containing index information are defined with a global scope, making them available to the MUF.
- The index phase must occur in MUF with full interaction with other tasks processing the index and sharing the same buffer pool. Each task does single block reads and writes.
- The area being reorganized cannot have other users for REORG to start and, once started, it cannot have other users to open.
- The MUF needs to be executing on the same image, that is, it cannot be accessed through XCF.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control.

Valid Entries:

YES or NO

Default Value:

NO

Note: The REORG MULTUSE=YES function requires that the DBUTLTY job run APF-authorized.

,FORMAT=

Specifies the load format, BACKUP or EXTRACT. The OPTION2= keyword must be specified as LOADONLY for FORMAT=EXTRACT to be valid.

Valid Entries:

BACKUP or EXTRACT

Default Value:

BACKUP

,KBYTES=

Specifies the number of 1024-byte blocks of memory each subtask acquires in order to write data rows. We recommend that you specify KBYTES=2000.

Valid Entries:

1 through 9999 (integers only)

This number, multiplied by 1024, is the number of bytes requested through GETMAIN.

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.

,NUMBER=

Specifies the number of contiguous tracks to consider as a group (or stripe). A group of tracks that is acquired for its use, by a subtask, are not available to any other subtask. This requires that the data area must have sufficient free space at completion of the load of the data to allow a partial group for every subtask. You should take this under consideration when when selecting the number of subtasks. Be aware that it is also required that at least one track be available for processing, not in a subtask group, once all the subtasks have completed.

Valid Entries:

Any desired number of tracks

Default Value:

15

,RESETRCE=

Specifies whether you want to keep the TSN (Transaction Sequence Number) that last updated this row. Specify NO to set the TSN to zero. Resetting the TSN to zero is only useful if the number is reset to a lower number during an initialization of the Log Area (LXX), or if the number wraps from FFFFFFFF to zero.

Valid Entries:

YES or NO

Default Value:

YES

,SLACK=

Specifies the number of bytes to reserve, during a load, in each data block, for compressed records to grow. The number entered applies only to blocks loaded to an area during this execution of the load. Blocks not loaded with data use the SLACK value specified in the AREA definition in CA Datacom Datadictionary.

Valid Entries:

A number that is less than the block size

Default Value:

0

,TABLE=

Specifies a table name to indicates that a single table is to be loaded. This parameter is required when FORMAT=EXTRACT is specified. It cannot be specified when FORMAT=BACKUP.

Valid Entries:

DATACOM-NAME of a table in the database and area being loaded

Default Value:

(No default)

,TASKS=

Specifies the number of subtasks that can be used to update the index.

Valid Entries:

1 through 25

Default Value:

10

Example JCL

Following is example JCL for a REORG BACKUPLOAD.

```
//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
//BACKUP1  DD DSN=cxx.bkup1,DISP=(,CATLG),         Output data
//BACKUP2  DD DSN=cxx.bkup2,DISP=(,CATLG),         Output data
//BACKUP3  DD DSN=cxx.bkup3,DISP=(,CATLG),         Output data
//                    UNIT=tape
//SYSIN             DD *                          Command Input
                    REORG OPTION2=BACKUPLOAD,AREA=A01,DBID=789,
                    INNUM=3,DDNAME=BACKUP,KBYTES=9999,NUMBER=1

/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
 REORG OPTION2=BACKUPLOAD,AREA=A01,DBID=789,
       INNUM=3,DDNAME=BACKUP,KBYTES=9999,NUMBER=1

FUNCTION=REORG
   AREA=A01
   DBID=00789
   DDNAME=BACKUP
   INNUM=003
   KBYTES=9999
   NUMBER=0001
   OPTION2=BACKUPLOAD
```

This sample page of a REORG BACKUPLOAD 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.

```
                    TABLE          RECORDS
                    C01               50
                    C02               60
                    F01               55
                    F02                0
                    F03                0
                    RNA                0
                    RNB                0
                    TOTAL            165

BACKUP FILE INFORMATION
JOBNAME  RUN UNIT    DATE       TIME    CXXNAME    DBID  SEQ  RECID  AREA
JOB1          nnn  mm/dd/ccyy hh.mm.ss  PRODCXX     789  NAT   NO    A01
JOB1          nnn  mm/dd/ccyy hh.mm.ss  PRODCXX     789  NAT   NO    A01
JOB1          nnn  mm/dd/ccyy hh.mm.ss  PRODCXX     789  NAT   NO    A01

INPUT FILE INFORMATION         OUTPUT TABLE INFORMATION
AREA  TABLE  ID     RECORDS     AREA  TABLE  ID     RECORDS     NOTES
A01   C01   101          50     A01   C01   101          50
A01   C02   102          60     A01   C02   102          60
A01   F01   103          55     A01   F01   103          55
A01   F02   104           0     A01   F02   104           0
A01   F03   105           0     A01   F03   105           0
A01   RNA   106           0     A01   RNA   106           0
A01   RNB   107           0     A01   RNB   107           0

OUTPUT AREA INFORMATION
          TOTAL       USED      TOTAL       USED PCNT    AVG     BLOCKS     BLOCKS       TOTAL
AREA     TRACKS     TRACKS     BLOCKS     BLOCKS FULL  SLACK   EXPANDED COMPRESSED     RECORDS
 A01         10          1        120         12   10     41        N/A        N/A         165
```

This page of the report shows the following:

TABLE RECORDS

Displays table and record information.

BACKUP FILE INFORMATION

Displays backup file information.

INPUT FILE INFORMATION

Displays input file information.

OUTPUT TABLE INFORMATION

Displays output table information.

OUTPUT AREA INFORMATION

Displays output area information.

Chapter 35: REPLACE (Replace All Data in a Table)

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 MUF and allows full read and update access to every other table in the same area and database to other applications in the same MUF. 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.

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 MUF 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 MUF 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 MUF, 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 MUF 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 MUF 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*

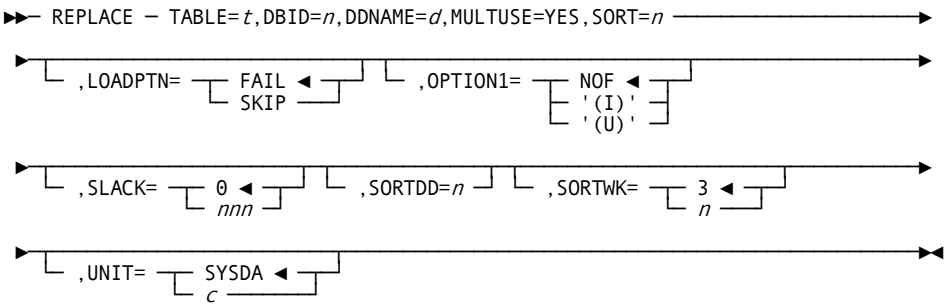
- The database may or may not be open for update in the connected MUF.
- The database may *not* be open for update in another MUF or Single User job.
- The database may *not* have any table in *unloading* status.
- The database ACCESS must be set OPTIMIZE and WRITE/UTLTY.
- The area ACCESS must be set WRITE/UTLTY.
- The table may not be in an open User Requirements Table.
- When MULTUSE=YES, this function will not be allowed against a Data Sharing MUF if more than one is executing.

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

- The database will be opened by MUF for update.
- The utility ACCESS AREA is not set (table level function not area level).

How to Use

The MUF must be active when you 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)

,DDNAME=

Specifies the JCL DDname for the input data set.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Valid Entries:

Any name valid to 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.

The OPTION1= parameter is highly recommended to obtain warning and critical messages if a problem occurs.

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

,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

,OPTION1=

Specifies the statistics display option to be passed to the sort program (see the SORT= parameter).

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

,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 Datadictionary.

Valid Entries:

A number that is less than the block size

Default Value:

0

SORTDD=

Specifies an optional DD statement, provided to the SORT that redirects SYSPRINT-type output from the SYSPRINT DD (used by DBUTLTY) to this optional DD statement.

Valid Entries:

printed output directed to optional DD statement

Default Value:

printed output directed to SYSPRINT, intermixed with DBUTLTY output

,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

,UNIT=

Specifies the unit name for sort work areas. The parameter is always edited, but is only used when the SORT= parameter is specified as not zero.

Valid Entries:

1- to 8-character unit name

Default Value:

SYSDA

Example JCL

Following is a JCL example for 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.

```
//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
//IN1      DD DSN=input                          Input data set
//*
//* NOTE: The following sort work allocations are not
//*       required if your sort package can dynamically
//*       allocate sort work areas.  Be sure to provide
//*       sufficient space for all keys to be sorted.
//*
//SORTWK01 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SORTWK02 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SORTWK03 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SYSIN    DD *                                  Command Input
                    REPLACE DBID=1,TABLE=F01,DDNAME=IN1,SLACK=12,
                    SORT=300,MULTUSE=YES

/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Following begins a REPLACE sample report.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8

  REPLACE DBID=1,TABLE=F01,DDNAME=IN1,SLACK=12,
          SORT=300,MULTUSE=YES

FUNCTION=REPLACE
   DBID=00001
   DDNAME=IN1
   MULTUSE=YES
   SLACK=0012
   SORT=000000300
   TABLE=F01
```

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.

```
                TABLE          RECORDS

                  F01               50
```

This page of the report shows information about the table and records.

Chapter 36: REPORT (Generate Reports)

The REPORT function produces the different types of reports described in the following sections.

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.

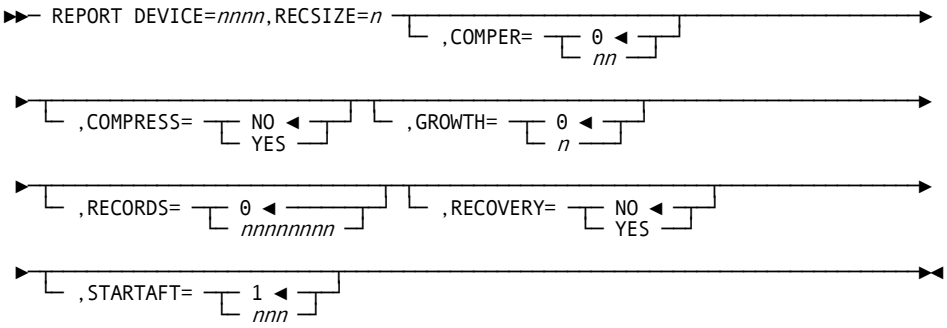
When to Use

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 with the MUF active. 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

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
        REPORT   DEVICE=3380,RECSIZE=400,COMPER=25,COMPRESS=YES,
                   GROWTH=25,RECORDS=250000,RECOVERY=YES
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT  DEVICE=3380,RECSIZE=400,COMPER=25,COMPRESS=YES,
              GROWTH=25,RECORDS=250000,RECOVERY=YES

FUNCTION=REPORT
   COMPER=25
   COMPRESS=YES
   DEVICE=3380
   GROWTH=025
   RECORDS=00250000
   RECOVERY=YES
   RECSIZE=00400
```

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.

```
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  #
         #-----------#-----------#-----------#-----------#-----------#
         #      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  #
         #-----------#-----------#-----------#-----------#-----------#
```

This page of the report shows a 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.

```
-------------------------------------------------------------------------
# DEVICE # RECORD # RECOVERY #  COMPRESSION  # FIRST #  RECORD  #  GROWTH #
#  TYPE  #  SIZE  #   Y/N    # Y/N # PERCENT # R P B # QUANTITY # PERCENT #
#--------#--------#----------#-----#---------#-------#----------#---------#
#  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 #
#   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 #
    .      .        .       .       .          .        .         .          .      .
    .      .        .       .       .          .        .         .          .      .
    .      .        .       .       .          .        .         .          .      .
```

This page of the report shows the parameters you coded with the REPORT DEVICE option. The report also displays the default values for parameters not coded.

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.

Directory (CXX) Reports

The REPORT function provides several types of reports containing information about the contents of 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.
- The Directory (CXX) Report with TYPE=I shows the impact report of the table status results of the last CA Datacom Datadictionary catalog of the database.
- The Directory (CXX) Report with TYPE=K shows the full CXX options in place and also provides information about encryption as defined to the CXX and tables.
- The Directory (CXX) Report with TYPE=P shows the current list of PREINIT data sets that are available for use during creation of a new:
  - database
  - index area
  - data area

Note: You can obtain most of the 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 MUF 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 for types A, B, and K. Type I requires the MUF to be enabled. For types A, B, and K the following apply:e .

If using Simplify mode and the MUF is enabled:

The MUF is used for all completely current information. An internal COMM STATS is generated for each data base reported to update the CXX.

If using Simplify mode and the MUF is down:

The report always expects to run with the MUF enabled. If not enabled, it expects to wait for the enable with the DELAY68 option of DBSIDPR. It may follow any previous function that runs with MUF only down or a function that sets execution to run without MUF using SET OPTION1= MUF_NOT_ENABLED. We do not recommend (but it is possible to use) function SET OPTION1=MUF_ENABLED_OR_DISABLED. If executed with the MUF down, the following section applies.

If not using Simplify mode or the MUF is down:

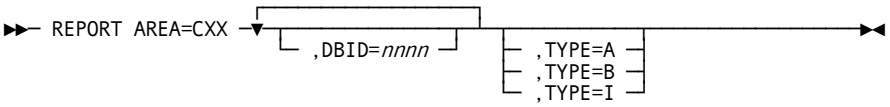
The Directory (CXX) reports read and reflect the information from the CXX data set. This information is correct when the database is not currently open by the MUF or any Single User jobs, including DBUTLTY. Even when it is open, the information is usually adequately current. If opened by the MUF, the 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 MUF.

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.

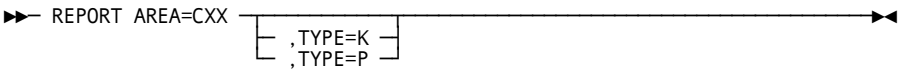
How to Use

You can execute this command in either Single User or with the MUF active.

Use the following control statement to request this report for types A, B, and I, with an optional DBID specification:



Use the following control statement to request this report for types K and P:



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 (without any TYPE= or with TYPE=K) has information about the full CXX.

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.

Note: The DBID is not allowed for TYPE=K and TYPE=P executions.

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 678).

,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 681).

,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 681).

,TYPE=K

Specifies the page one of a full CXX report (no TYPE= or TYPE=B) be produced with the addition of any Encryption information provided for the CXX in general and tables with any form of Encryption specified. For more information on the TYPE=K report, see Type K - Encryption Report (see page 683).

,TYPE=P

Specifies the PREINIT data set report be generated. For more information on the TYPE=P report, see Type P - PREINIT Report (see page 683).

Full Directory Report

To obtain current status and control information, generate a full Directory Report.

Example JCL: Full CXX Report

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.

```
//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   AREA=CXX
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

If a report is done without a DBID, the first page of the CXX report (without any TYPE= or with TYPE=K) has information about the full CXX. In the following example, a full CXX Report (without a DBID specified) displays the report. This same report information is provided with the TYPE=K report, except the TYPE=K report includes any encryption information (not provided below).

```
CONTROL AREA DIRECTORY                                CXX VERSION - 14.0
DBCRBAS REQUIREMENT (ALL BASES OPEN)  -    76,728     CXX ENQ - SYSPLEX
NUMBER OF DATA BASES -    32                          DATA HIGH USED MARK - YES
DEVICE TYPE - 3390                                    DATASET EXTENT VALIDATION - YES
DYNAMIC FILE ALLOCATION ALLOWED - YES                 DATA FAST SEARCH - YES
SECURE USING JOBNAME - NO                             CXX LEVEL -   1
SECURE ALLOWING SINGLE USER - NO                      SINGLE USER ALLOWED - NO
SQL MODE - SUPPORT ACTIVE - YES                       SIMPLIFY MODE - YES
           DATADICTIONARY BASE -     2
           DATA DEFINITION DIRECTORY BASE -     15    PREINIT AREAS - YES
```

The 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.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    REPORT AREA=CXX,DBID=999

                    FUNCTION=REPORT
                       AREA=CXX
                       DBID=00999
```

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.

```
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--mm/dd/ccyy, TIME-hh.mm.ss
LAST RESTARTED  - DATE--mm/dd/ccyy, TIME-hh.mm.ss
LAST MAINTAINED - DATE--mm/dd/ccyy, TIME-hh.mm.ss
REPLICATED BASE  -  NO                                  BASE FORMAT -   2
PARTITIONED BASE -  NO
UPDATE USERS    -   0
READ ONLY USERS -   0
OCCURRENCE - TOBIE_DATABASE_PTN P001
SQL SECURITY -     NO

INDEX INFORMATION  - IXX
DEVICE TYPE  -      3390                          DYNAMIC EXTEND -    YES
BLOCK LENGTH -     4,096                             DYN.EXT.TRACKS    15
LEVELS          -      1                             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
```

This page of the report shows the following:

Database Information

- 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.
- 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.

- 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.

- REPLICATED and PARTITIONED BASE indicate whether the database is CA Datacom STAR replicated or partitioned. (See the *CA Datacom STAR Administration Guide* for details on partitioning and replication.)
- BASE FORMAT is a number indicating feature support. The number displayed can be a 1 or a 2.
- UPDATE USERS reports if the base is currently owned (locked) by a copy of CA Datacom. A zero number means not owned, a not zero is owned. Without Simplify mode, the only number used is a 1. With Simplify, it may be a 2, indicating Simplify owned by DBUTLTY; a 3, indicating Simplify owned by MUF not for Single User DBUTLTY functions; or a 4, indicating Simplify owned by MUF for Single User DBUTLTY functions.
- READ ONLY USERS reports zero (none) or one (one or more) if the database is opened in MUF without ACCESS=OPTIMIZE and all open URTs specified UPDATE=NO.
- OCCURRENCE is the entity-occurrence name for the database, and the status/version in CA Datacom Datadictionary.
- 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.

- 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.
- 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.
- 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.
- These lines report the space utilization for this Index Area:

  BLOCKS /TRACK

  Number of Index blocks per track.

  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.
- 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 by issuing a catalog locate. If the locate fails, the locate return code is displayed.
- DYNAMIC EXTEND is specified as YES or NO for the Index Area.

- 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.

  Note: If DYNAMIC EXTEND is specified as YES and this is zero, the VTOC secondary allocation is used.

- CA Datacom has taken advantage of enhancements to z/OS environments to allow dynamic extend to be greater than 64K tracks.

- 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.

- 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.

- 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.

```
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  ENCRYPTION - NONE                 DEVICE TYPE  -    3390
LOGGING  - YES                                       BLOCKS/TRACK -   12
PIPELINE - YES                                     DSOP MODE    - NO RECLAMATION
DB COMPRESSION   - NO                             TABLE LOADED - YES, DATE--mm/dd/ccyy, TIME-hh.mm.ss
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 URI-REUSE NO
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 - JBND007.P025000                      DATA  DTF/DDNAME - P025000
VOLUMES       - WRKD32
KEY - NAME   ID   LENGTH  MASTER  NATIVE  UNIQUE  DIRECT  KEY INC  KEY USE  OCCURRENCE  /  KEY SQLNAME           STATUS
*     TST01  0001      8   YES     YES     NO       NO      YES      ANY    TST01P001                           LD
                                                                            TOBIE_K01_PTN_TST01
```

This page of the report displays the following information:

Following is table information:

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 Datadictionary entity-occurrence name for the table and its status/version.

TABLE ID

Internal table identifier.

Following 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 Datadictionary 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.

Following 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 MUF 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.

Following 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.

High Used Track

Specifies the number of tracks which must be processed to ensure all data is read during the execution of a physical backup or GETPS. It is also the minimum number of tracks to format by the next load, and the minimum "move to" data set size after dynamic extend for an ONLINE_AREA_MOVE. NONE indicates either the high used mark option is set to off or the area has not been reloaded since it was turned on.

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*.

Note: If DYNAMIC EXTEND is specified as YES and this is zero, the VTOC secondary allocation is used.

CA Datacom has taken advantage of enhancements to z/OS environments to allow dynamic extend to be greater than 64K tracks.

CHANGE MASTER KEY and DUPLICATE MASTER KEY

Report whether the Master Key value can be changed during update processing and whether duplicate Master Key values are allowed.

RECOVER and LOGGING

Rport whether CA Datacom/DB recovery is activated for this table and whether logging is active for this table.

ENCRYPTION

Is either NONE or the mode and type of encryption, such as (B(BASIC) A(AES128).

PIPELINE

Indicates whether the Pipeline maintenance option is active for this table.

DSOP MODE

Rports the data area space reclamation option for this table.

DB COMPRESSION and USER COMPRESSION

Indicate 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.

The fllowing 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.

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 Datadictionary 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.'

DATADICTIONARY BASE

the database ID of the CA Datacom Datadictionary where the definitions were stored when this table was cataloged to the Directory.

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=.

DEFINITION POSTED VIA

Indicates if the table was posted to the Directory using the CA Datacom Datadictionary CATALOG, the (DDCFBLD -CXX BUILD transaction and the CXXMAINT DDPROD option), or LOAD FROM BACKUP.

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.

URI-REUSE

Indicates whether the CA Datacom Datadictionary attribute URI-REUSE is set to yes or no. When URI-REUSE is set to Y (yes), when the URI count reaches 3G, an asynchronous scan of the data area is scheduled in the MUF. For more information on URIs and the URI reuse feature, see the *CA Datacom/DB Database and System Administration Guide*.

Following 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.

Following 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.

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.

Following 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 SET NAME

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 by issuing a catalog locate. If the locate fails, the locate return code is displayed.

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.

Following is report of 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.

```
TABLE K12 CONTINUED

    - FIELDS - ORDER    SENSITIVE  NULL  START  LENGTH  OCCURRENCE
               ASCEND   NOT SENS   NO       28       8  FIRST-RANDOM-NO P002
                 .
                 .
                 .

ELEMENTS - NAME   SECURITY CODE   START  LENGTH   DATA TYPE  NULL COL.  OCCURRENCE
           TSTEL     00-000          0      84   CHARACTER     YES     TSTEL P002
```

This page of the report shows the following:

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.

- 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.

```
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
```

This page of the report shows the following:

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.

```
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
```

This page of the report shows 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
- SYSTEM - Indicates an accessor ID default (default system user)
- 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.

Encryption Report without TYPE=A

Table information relating to encryption is provided as part of the full REPORT AREA=CXX without TYPE=A. For an example, see Sample Report: ENCRYPT (see page 424).

Example JCL: Space Utilization CXX Report

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.

```
//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   AREA=CXX,TYPE=A
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT  AREA=CXX,TYPE=A

FUNCTION=REPORT
   AREA=CXX
   TYPE=A
```

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.

```
                          DATA AREA SPACE UTILIZATION REPORT


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
  .    .           .            .           .          .     .      .           .    .
  .    .           .            .           .          .     .      .           .    .
  .    .           .            .           .          .     .      .           .    .
```

This page of the report provides the following information:

AREA NAME

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.

TOTAL TRACKS

The number of tracks allocated and accepted by CA Datacom/DB for use in the area. 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.

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.

TOTAL BLOCKS

Number of blocks in the area.

USED BLOCKS

Number of blocks that contain data. If a system failure has occurred during maintenance processing, this number can be inaccurate.

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.

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.

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 Datadictionary 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 Datadictionary CATALOG, the reasons are printed on the report. For more information and samples, see CXXMAINT DDPROD (Update Directory) (see page 365).

Impact CXX Report

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.

```
//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 AREA=CXX,TYPE=I,DBID=16
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                CONTROL CARD(S)
                .........1.........2.........3.........4.........5.........6.........7.........8
                REPORT AREA=CXX,TYPE=I,DBID=16

                FUNCTION=REPORT
                   AREA=CXX
                   TYPE=I
                   DBID=16
```

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.

```
                              IMPACT REPORT
BASE        AREA        TABLE ID        DISPOSITION
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)
```

This page of the report shows the following:

BASE

The DATACOM-ID of the database containing the area.

AREA

The DATACOM-NAME of the area.

TABLE

The table name.

ID

The table ID.

DISPOSITION

The reason is described in Processing a DDPROD Request (see page 367).

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. 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 r11, that is, tables whose disposition is LOADED have no need to also show the key-level status.

The code is described in Processing a DDPROD Request (see page 367).

Type K - Encryption Report

For information about the AREA=CXX,TYPE=K encryption information report, see Sample Report: ENCRYPT (see page 424).

Type P - PREINIT Report

Use REPORT AREA=CXX with TYPE=P when you want to request a Directory PREINIT Report.

Example JCL

The following shows the command to generate a TYPE P (PREINIT) CXX 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.

```
//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 AREA=CXX,TYPE=P
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

The PREINIT list exists at the CXX level and is reported in a REPORT function requested with AREA=CXX,TYPE=P.

```
                CONTROL CARD(S)
                .........1.........2.........3.........4.........5.........6.........7.........8
                REPORT AREA=CXX,TYPE=P

                FUNCTION=REPORT
                   AREA=CXX
                   TYPE=P
```

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.

```
DBID AREA TYPE  BLKSIZE OLDEP  DATETIME            TRACKS  DSN

 995 IXX INDEX    4,096    NO  20131106084732          15  SYS13310.T084731.RA000.DBDVLWXX.IXX997.H02
 997 I11 INDEX    4,096   YES  20131106084731          15  SYS13310.T084731.RA000.DBDVLWXX.I11997.H02
 998 I11 INDEX    4,096    NO  20131106084731          15  SYS13310.T084731.RA000.DBDVLWXX.I11997.H02
 999 I11 INDEX    8,192   YES  20131106084731          15  SYS13310.T084731.RA000.DBDVLWXX.I11997.H02
1555 ABC  DATA    5,555   N/A  20131106084732          15  SYS13310.T084731.RA000.DBDVLWXX.ABC555.H02
3333 ABC  DATA    4,096   N/A  20130628091120          15  SYS13179.T091119.RA000.DBDVLWXX.ABC.H04
3333 IXX INDEX    4,096   YES  20130628091120          15  SYS13179.T091119.RA000.DBDVLWXX.IXX.H04
```

The section from a full CXX Report that follows shows an entry for PREINIT AREAS at the bottom right. PREINIT AREAS - YES indicates that currently there are PREINIT areas, but PREINIT AREAS - NO indicates that there are currently no PREINIT areas.

```
CONTROL AREA DIRECTORY                                  CXX VERSION - 14.0
DBCRBAS REQUIREMENT (ALL BASES OPEN)  -   105,616      CXX ENQ - SYSPLEX
NUMBER OF DATA BASES -    45                            DATA HIGH USED MARK - YES
DEVICE TYPE - 3390                                      DATASET EXTENT VALIDATION - YES
DYNAMIC FILE ALLOCATION ALLOWED - YES                   DATA FAST SEARCH - YES
SECURE USING JOBNAME - NO                               CXX LEVEL -   1
SECURE ALLOWING SINGLE USER - NO                        SINGLE USER ALLOWED - YES
SQL MODE - SUPPORT ACTIVE - YES                         SIMPLIFY MODE - YES
           DATADICTIONARY BASE -     2
           DATA DEFINITION DIRECTORY BASE -    15       PREINIT AREAS - YES
```

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 or Inn). You can request this report in either table name (TYPE=A) or key ID (TYPE=B) sequence. The statistics from which these reports derive the data are stored in the Directory (CXX). Therefore, if you move or copy the Index for a database using non-CA Datacom utilities (that is, DFDSS), and if you do not use the DBUTLTY function CXXCLONE (or DBUTLTY function LOAD AREA=CXX) to restore the CXX definitions and statistics into the target CXX, these statistics are not be accurate. The counts are likely to be inaccurate after a MUF outage other than an EOJ.

  Take a backup of the source CXX first. Following is an example of the CXXCLONE statement:

  ```
  CXXCLONE DBID=1022,STATUS=NO_CHANGE,DDNAME=BKPCXX,OPTION=DELETE
  ```

- 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 only if requested by CA Support.
- The number of records that have unique values for key fields (TYPE=G).

Note: TYPE=A,B,D,E, and G run only in MUF, providing 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.

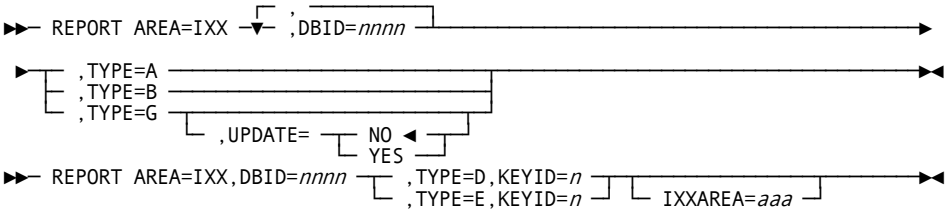
The statistics reported in a type A or B Report are updated when the data is loaded, when the Index is rebuilt, or by using the CXXCLONE DBUTLTY function or LOAD AREA=CXX.

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 CA Support only.

How to Use

You can execute this command in either Single User or with the MUF active. 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 CA 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 MUF and can occur while any other MUF 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.

```
//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   AREA=IXX,DBID=1,TYPE=A
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Sample Report: REPORT IXX - Type A

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT  AREA=IXX,DBID=1,TYPE=A

FUNCTION=REPORT
   AREA=IXX
   DBID=1
   TYPE=A
```

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.

The spacing in the example is adjusted for display purposes.

```
INDEX  POPULATION  REPORT  -  BY TABLE

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
```

This page of the report displays the following information:

TABLE

The DATACOM-NAME of the table.

KEY ID

The DATACOM-ID of the key.

KEY NAME

The DATACOM-NAME of the key.

RECORDS

Number of records indexed by this key. If a system failure has occurred during maintenance processing, this number can be inaccurate.

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, 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.

```
//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   AREA=IXX,DBID=1,TYPE=B
/*
```

Sample Report: REPORT IXX - Type B

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT  AREA=IXX,DBID=1,TYPE=B

FUNCTION=REPORT
   AREA=IXX
   DBID=1
   TYPE=B
```

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.

```
INDEX  POPULATION  REPORT  -  BY  KEY  ID


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
```

This page of the report displays the following information:

KEY ID

The DATACOM-ID of the key.

TABLE

The DATACOM-NAME of the table containing the key.

KEY NAME

The DATACOM-NAME of the key.

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.

```
//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   AREA=IXX,DBID=1,TYPE=D,KEYID=1
/*
```

Sample Report: REPORT IXX - Type D

```
CONTROL CARD(S)
........1.........2.........3.........4.........5.........6.........7.........8
REPORT  AREA=IXX,DBID=1,KEYID=1,TYPE=D

FUNCTION=REPORT
   AREA=IXX
   DBID=1
   KEYID=1
   TYPE=D
```

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.

```
                                                                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*
```

This page of the report is normally followed by several more pages.

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.

```
//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   AREA=IXX,DBID=1,TYPE=G,UPDATE=YES
/*
```

Sample Report: REPORT IXX - Type G

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT  AREA=IXX,DBID=1,TYPE=G,UPDATE=YES

FUNCTION=REPORT
   AREA=IXX
   DBID=1
   UPDATE=YES
   TYPE=G
```

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.

```
INDEX  POPULATION  REPORT  -  BY TABLE

                                          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
```

This page of the report displays the following information:

TABLE

The DATACOM-NAME of the table.

KEY ID

The DATACOM-ID of the key.

KEY NAME

The DATACOM-NAME of the key.

TOTAL RECORDS

Number of records indexed by this key at LOAD time, RETIX time, or currently, if UPDATE=YES is specified.

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:
  1st value       LNAME MI FNAME ADDR

  2nd value       LNAME MI FNAME ADDR

  3rd value       LNAME MI FNAME ADDR

  4th value       LNAME MI FNAME ADDR
```

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.

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 MUF 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

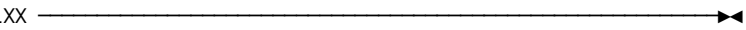
Use the REPORT AREA=LXX option to obtain information about the Log Area, specifically you can use this report to determine the highest Transaction Sequence Number (TSN).

In Simplify mode, this report expects to connect to MUF and get the current information from MUF. It may only execute without MUF if it follows another function that only runs with MUF not enabled or if the MUF is acknowledged as down using function SET OPTION1=MUF_NOT_ENABLED. It is possible but not recommended to use SET OPTION1=MUF_ENABLED_OR_DISABLED. This report may not follow in the same DBUTLTY step with a previous function that opens the LXX. If needed, it must be in a different step.

Without Simplify mode you can request a Log Area report any time. The information obtained reflects information stored on disk, not necessarily the current information the MUF is using.

How to Use

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.

```
//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
//LXX      DD DSN=lxx.data.set,DISP=SHR          Log Area data set
//SYSIN    DD *                                  Command Input
         REPORT   AREA=LXX
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
 CONTROL CARD(S)
 .........1.........2.........3.........4.........5.........6.........7.........8
REPORT AREA=LXX

 FUNCTION=REPORT
    AREA=LXX
```

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.

```
                              LOG FILE (LXX) OUTPUT
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
```

This page of the report shows messages that indicate the status of the Log Area.

LOG FILE HIGHEST TRANSACTION SEQUENCE NUMBER

Displays the last transaction sequence number in the Log Area.

LOG VARIABLE

Indicates if the LXX is initialized as variable (YES) or not (NO), specified by the DBUTLTY INIT AREA=LXX VARIABLE= option.

date and time

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. It is printed as binary zeros if the LXX or FXX was not subject to INIT in r11 SP2 or later. Following the date and time is the store clock value (in hex, *cccccccccccccccc*, 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 (*cccccccccccccccc*) at the time of the RESET.

FXX TOTAL TASKS

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.

FXX USER TASKS

Displays 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.

LAST LOG RECORD

Provides the date and time (*ccyymmddhhmmss*) and store clock value (*cccccccccccccccc*) 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).

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)

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.

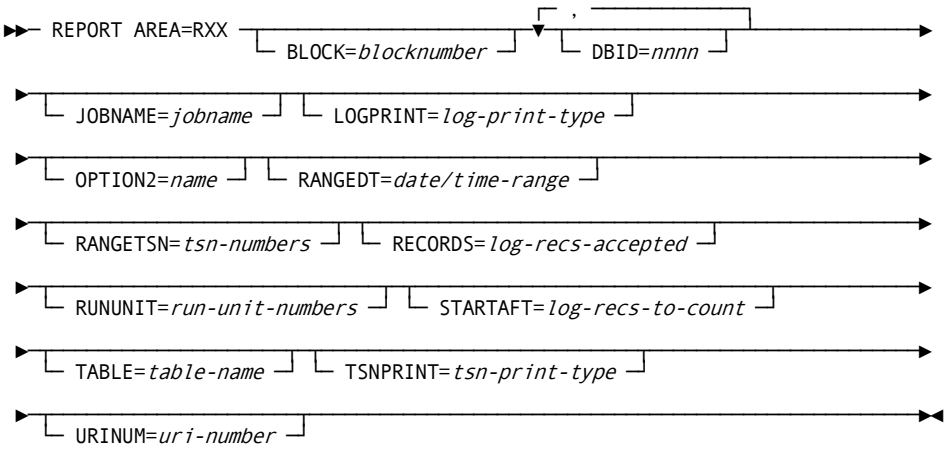
This utility always reads one input file of one or more volumes. In z/OS, this can be multiple concatenated files which appear to CA Datacom/DB as one file. In all environments, if the input is not log data, unexpected results occur. z/OS users can run the report against the Log Area (LXX) directly. This is not recommended, but can be requested by CA Support. If this is done, one or more of the various warnings is to be expected. This is because:

- The Log Area is being actively changed,
- The Log Area is pipelined in MUF memory, and
- The Log Area is not treated as a perfectly ordered sequential file.

The REPORT function, like all DBUTLTY functions, is designed and expected to run authorized in z/OS. If the RXX was built from an LXX that was ever used with the MUF option LOG_RECORD_ENCRYPT set to YES since the LXX was last initialized or reset, the REPORT function is required to run authorized. It does not matter if the RXX being read has encrypted data. Potentially authorized data is not intended to be in an unauthorized address space.

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 Recovery file (split), the Report control statements are as shown in the Split File Analysis Report syntax on page Split File Analysis Report.



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 Keywords

AREA=RXX

Requests the Recovery File Analysis Report.

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 CA 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 Recovery File (RXX) Analysis Report. Specifying SUMMARY produces the basic transaction report. For more information and examples see TSNPRINT=DETAIL Sample (see page 710).

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=4M
//STEPLIB    See the previous note and JCL Requirements.
//RXX      DD DSN=rxx.data.set,DISP=SHR          RXX data set
//SYSOUT   DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SNAPER   DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//*
//SYSIN    DD *                                  Command Input
      REPORT AREA=RXX,LOGPRINT=DETAIL,TSNPRINT=DETAIL,DBID=015,TABLE=MSG
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Note: The following sample RXX Analysis Report is for a database with an ID of 15 and a table named MSG, using LOGPRINT=DETAIL and TSNPRINT=DETAIL.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT AREA=RXX,LOGPRINT=DETAIL,TSNPRINT=DETAIL,TABLE=MSG,DBID=15

FUNCTION=REPORT
   AREA=RXX
   DBID=00015
   LOGPRINT=DETAIL
   TABLE=MSG
   TSNPRINT=DETAIL
```

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.

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
      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
         .
         .
         .
```

This portion of the report displays the following information:

CMD

Displays the 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 typically 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 Datadictionary 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

Displays the 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

Displays the Database ID. Blank for records without database ID's. Printed as a decimal number.

URI

Displays the Unique Row Identifier. Blank for records without a URI number. Printed in hexadecimal format. Contains the literal 'NON-URI ' for areas not loaded as URI.

BLOCK

Displays the data area block number. Blank for records without a data block. Printed in hexadecimal format.

L#

Displays the 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

Displays the Transaction Sequence Number. Blank for records not part of a transaction. Printed in hexadecimal format.

JOB NAME

Displays the job name.

R-UNIT

Displays the run unit number.

MP

Blank if not running r10 or later as a MUFplex. Otherwise, contains a MUFplex number of which MUF logged this log record.

USER ID

Displays the User ID. First 8 characters of the 32-byte user ID passed on the request.

AREA

Displays the data area name containing row.

MONITOR INFO

Displays the monitor information, if this request was made through the CA Datacom CICS Services or the CA Datacom Server.

INPUT RECORD

Displays the number of this input log record.

FPR

Displays the 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

Displays the date the log record was added. Printed in the form mm/dd/ccyy.

TIME

Displays the time the log record was added. Printed in the form HH.MM.SS.

CLOCK

Displays the result of a hardware STCK (store clock) instruction at the time the log record was added.

SEC.USER

Displays the security user requesting maintenance.

MOVE TO

Displays the block number to where an update moved a record. Printed in hexadecimal format.

LRID

Displays the 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 CA Support, should the need arise. Printed in hexadecimal format.

PREV

Displays the previous log record ID. Log records for a transaction are chained in reverse back to a previous one of low values.

LRSN

Displays the 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 CA Support, should the need arise. The number is printed in hexadecimal.

LBSN

Displays the log block sequence number. An internal number used by CA 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 CA Support, should the need arise. The number is printed in hexadecimal.

TASK

Displays the number of the task area which was used when this log record was generated.

STATUS

Displays the 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 roll back.
- **B** - Specifies that this record is subject to transaction backout.
- **C** - Specifies a log record subject to Change Data Capture.
- **F** - Specifies that the first log record after transaction was force spilled.
- **M** - Specifies an update command caused the record to move.
- **T** - Specifies a log record that is part of a transaction that has at least one record subject to Change Data Capture.
- **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 CA Support and are not always present even though the description would indicate that it should be present.

RECL

Displays the size of the log record. It is printed in hexadecimal. This field is provided to assist CA Support, should the need arise.

DATA

Displays the size of the log data in this log block. It is printed in hexadecimal. This field is provided to assist CA 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

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
```

This portion of the report displays TSN Detail Report section.

TSN

Displays the transaction sequence number.

JOB

Displays the name of the job.

FIRST RECORD

Displays the date and time the first record in a transaction was processed.

LAST RECORD

Displays the date and time the last record in a transaction was processed.

REQUESTS

Displays 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

Displays the name of the table associated with the transaction.

DBID

Displays the database ID associated with the transaction.

ADD

Displays the total number of add requests for a transaction.

DELETE

Displays the total number of delete requests for a transaction.

UPDATE

Displays 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.

```
     REQUEST SUMMARY REPORT
COMMAND  TABLE DBID AREA  OCCURRENCES PERCENT

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
MOVERU   C02    997 A01        1,056      1.0
UPDAT    C02    997 A01       11,355     14.2

ADDIT    TOTAL                 38,034     36.4
DELET    TOTAL                 28,115     27.0
UPDAT    TOTAL                 28,056     35.1
```

PERCENT

The percentage is the percent of the occurrence records for the entire RXX input. Restrictive options like DBID= do not change the percent.

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.

ADDIT DELET UPDAT

The summary section displays 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.

A summary line for each of ADDIT, UPDAT, and DELET shows the total records and their percent.

Additional Summary Information

```
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
```

This page of the report shows an RXX Analysis Report that provides more summary information as follows.

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 numbers 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 CA 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

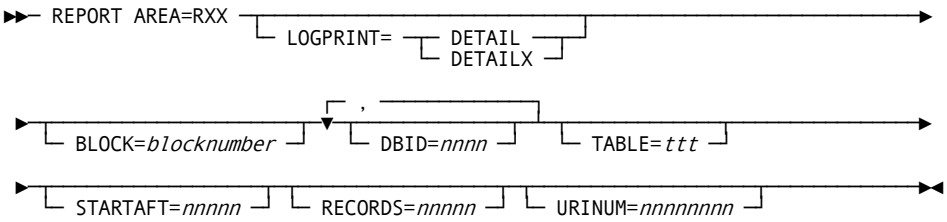
The Recovery File Analysis Report program can detect the type of input file. If the input is a CA Datacom Fast Recovery file (split), the report control statements are as shown in the following syntax diagram.

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)
- LOGPRINT=DETAILX Example Split File Report (see LOGPRINT=DETAILX Example Split File Report)

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 CA Support.

DBID=

Use to specify a database ID. 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 (a range, for example, 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.

LOGPRINT=DETAILX prints the entire record in hexadecimal. For an example report generated with LOGPRINT=DETAILX, see LOGPRINT=DETAILX Example Split File Report.

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. In z/VSE, the file must be on tape/cart and not DASD. If the input is not log data, unexpected results occur.

Following is an example report generated by specifying LOGPRINT=DETAIL (same as the basic syntax without options, that is, REPORT AREA=RXX).

```
                CONTROL CARD(S)
                .........1.........2.........3.........4.........5.........6.........7.........8
                REPORT AREA=RXX

                FUNCTION=REPORT
                   AREA=RXX
```

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.

```
SPLIT FILE      GROUP         GROUP START              GROUP END
                GROUP9     1/05/00,10.40.44         1/05/00,10.41.45
     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
                     .
                     .
                     .
```

This page of the report shows the following:

GROUP

Identifies the split file group.

GROUP START

Displays the date and time the group started.

GROUP END

Displays the date and time the group ended.

DBID

Displays a database ID. This is left blank for records without database IDs. It is printed as a decimal number.

TABLE

Displays a database 3-character table name.

URI

Displays a Unique Row Identifier. This is left blank for records without a URI number. It is printed in hexadecimal format.

BLOCK

Displays a data area block number. This is left blank for records without a data block. It is printed in hexadecimal format.

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 CA 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 the following:

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, 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).

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT AREA=RXX,LOGPRINT=DETAILX

FUNCTION=REPORT
   AREA=RXX
   LOGPRINT=DETAILX
```

The following example report generated by specifying LOGPRINT=DETAILX (to give a hexadecimal dump of the entire record) 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.

```
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                                                              *                                 *

     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           *
                    .
                    .
                    .
```

This page of the report shows the following:

GROUP

Identifies the split file group.

GROUP START

Displays the date and time the group started.

GROUP END

Displays the date and time the group ended.

DBID

Displays a database ID. This is left blank for records without database IDs. It is printed as a decimal number.

TABLE

Displays a database 3-character table name.

URI

Displays a Unique Row Identifier. This is left blank for records without a URI number. It is printed in hexadecimal format.

BLOCK

Displays a data area block number. This is left blank for records without a data block. It is printed in hexadecimal format.

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 CA 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 the following:

A

Add

D

Delete

M

Moved

U

Update

R

Rollback

I

Incomplete task

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 MUF. 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= (see page 721) in the list of Required Keywords.

Restrictions

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

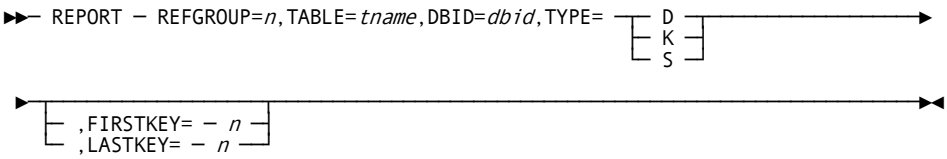
The DBUTLTY external resource name for the REFGROUP report is REPORT.REFGROUP. At the table level it uses the read right.

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 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
       REPORT REFGROUP=2,DBID=2,TABLE=ALS,TYPE=S
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Sample Report: REFGROUP Summary

```
                    CONTROL CARD(S)

.........1.........2.........3.........4.........5.........6.........7.........8
                    REPORT   REFGROUP=2,DBID=2,TABLE=ALS,TYPE=S
                    FUNCTION=REPORT
                       DBID=00002
                       REFGROUP=00002
                       TABLE=ALS
                       TYPE=S
```

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.

```
                    Table Block Usage Report by Reference Group
              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
```

This page of the report shows the following:

Groups

(698181 in this example) Displays the total number of REFGROUPs for this table, based on the value coded for REFGROUP.

Rows

(12,499,845 in this example) Displays the total rows in the table.

Rows/Blk

(8 in this example) Displays the average rows within a REFGROUP per block.

1 Row

(609,196 in this example) Displays the total number of data blocks with only 1 row for a REFGROUP.

2 Rows

(224,261 in this example) Displays the total number of data blocks with 2 rows for a REFGROUP.

3 Rows

(71,764 in this example) Displays the total number of data blocks with 3 rows for a REFGROUP, and so on through 9 Rows (11,804 in this example), the total number of data blocks with 9 rows for a REFGROUP.

The remainder of the Rows descriptions are as follows:

10+ Rows

(43,569 in this example) Displays the total number of data blocks with 10 through 14 rows for a REFGROUP.

15+ Rows

(331,738 in this example) Displays 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) Displays the total number of data block with 30 through 39 rows for a REFGROUP.

40+ Rows

(7,293 in this example) Displays the total number of data block with 40 through 49 rows for a REFGROUP.

50+ Rows

(blank in this example) Displays the total number of data block with 50 or more rows for a REFGROUP.

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
       REPORT REFGROUP=2,TABLE=FLD,DBID=2,TYPE=D
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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.

```
                  CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
                   REPORT   REFGROUP=2,DBID=2,TABLE=FLD,TYPE=D
                   FUNCTION=REPORT
                      DBID=00002
                      REFGROUP=00002
                      TABLE=FLD
                      TYPE=D
```

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.

```
                        Table Block Usage Report by Reference Group
  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

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
```

This page of the report shows the following:

Hex Group

Displays an ascending counter starting with 1 assigned to each REFRGOUP. It is a hexadecimal number.

Rows

Displays the total number of rows in this REFGROUP.

Rows/Blk

Displays the average rows per block for this REFGROUP.

Ref Blks

Displays the blocks in this REFGROUP. This is the same as the value coded for REFGROUP, except for the last REFGROUP.

1 Row through 50+ Rows

Displays the same explanation as was provided in the TYPE=S REFGROUP report, except that the TYPE=D REFGROUP report is reported for each REFGROUP.

At the end of the TYPE=D REFGROUP report are the totals that summarize the report.

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
       REPORT REFGROUP=2,DBID=2,TABLE=FLD,TYPE=K
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

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.

```
                  CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
                    REPORT   REFGROUP=2,DBID=2,TABLE=FLD,TYPE=K
                    FUNCTION=REPORT
                       DBID=00002
                       REFGROUP=00002
                       TABLE=FLD
                       TYPE=K
```

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.

```
                  Table Block Usage Report by Reference Group

  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

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
```

This page of the report displays the following information:

HEX GROUP

Displays an ascending counter starting with 1 assigned to each REFRGOUP. It is a hexadecimal number.

ROWS

Displays the total number of rows in this REFGROUP.

ROWS/BLK

Displays the average rows per block for this REFGROUP.

REF BLKS

Ref Blks shows the blocks in this REFGROUP. This is the same as the value coded for REFGROUP, except for the last REFGROUP.

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.

GROUPS and TOTALS

At the end of the TYPE=D REFGROUP report are the totals that summarize the report.

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.

```
                  Key Values by Reference Group
Group  First Key Value

    1  FLDACCTS
       CDCCCCEE4444444444444444444444444400004444444444444444444
       6341333200000000000000000000000000010000000000000000000000
    2  FLDAGGREGATE                          AGR-REQRD
       CDCCCCDCCCEC444444444444444444444444000FCCD6DCDDC44444444444
       6341779571350000000000000000000000062C1790958940000000000
    3  FLDDATABASE                                DATABASE-USER
       CDCCCECCCEC4444444444444444444444440009CCECCCEC6EECD4444444
       6344131212500000000000000000000000000C13413121250425900000000
    4  FLDDDMT3002
       CDCCCDEFFFF4444444444444444444444440000444444444444444444444
       6344443300200000000000000000000000010A00000000000000000000
    5  FLDDDMT5100
       CDCCCDEFFFF4444444444444444444444440000444444444444444444444
       6344443510000000000000000000000000030000000000000000000000
    6  FLDDIR_DATASET
       CDCCCD6CCECECE444444444444444444444000544444444444444444444
       634499D413125300000000000000000000001030000000000000000000
    7  FLDFIELD
       CDCCCCDC444444444444444444444444444010D44444444444444444444
       6346953400000000000000000000000000011C00000000000000000000
             .
             .
             .
```

The 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.

REPORT MEMORY=CF (Coupling Facility)

In a Data Sharing environment, REPORT MEMORY=CF,DDNAME=nnnnnnnn (,DUMPS=FULL) can be used to produce a report of list structure data and locks, where DDNAME=nnnnnnnn specifies a structure name within the MUFplex (the name specified in the MUFPLEX MUF startup option should be used). A DUMPS=FULL option is available to include in the dump all of the index and data buffers in the Coupling Facility. All list structure records are dumped and all list locks are reflected as available on whichever MUF owns them. See the *CA Datacom/DB Database and System Administration Guide* for more information on Data Sharing.

Important! The MEMORY=CF report is provided for use by CA 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.

Note: The REPORT MEMORT=CF function requires that the DBUTLTY job run APF-authorized.

REPORT MEMORY=DS (Data Space)

At the direction of CA Support, REPORT MEMORY=DS can be used to obtain a dump of the contents of a data space. To use MEMORY=DS, the MUF must be enabled and the data space active.

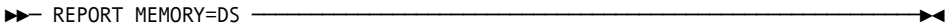
Note: The REPORT MEMORY=DS function requires that the DBUTLTY job run APF-authorized.

When to Use

You can generate this report at any time.

How to Use

You can execute this command in either Single User or with the MUF active. The following is the control statement to request this report:



Command

REPORT

Invokes the function to produce a report.

Required Keywords

MEMORY=DS

Requests the dump of the data space.

Example JCL

The following shows the command to generate the dump of the data 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.

```
//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   MEMORY=DS
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                CONTROL CARD(S)
                .........1.........2.........3.........4.........5.........6.........7.........8L
              REPORT MEMORY=DS

                FUNCTION=REPORT
                 MEMORY=DS
```

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.

```
JOB MEMDS           STEP TEST           TIME 082458   DATE 01263    ID = 020    CPUID = 00140AA12064   PAGE 00000001

  PSW AT ENTRY TO SNAP    078D5000  80054EC2  ILC  02  INTC  0033
DATA SPACE STORAGE
STORAGE FOR DATA SPACE 00000DB   ASID C3
              .
              .
              .
     LINES 0000A840-0000F3E0  SAME AS ABOVE
     LINES 0000F440-00013FE0  SAME AS ABOVE
     LINES 00014040-00018BE0  SAME AS ABOVE
     LINES 00018C40-00018FE0  SAME AS ABOVE
```

- This page of the report shows that the data space dump begins and is summarized at the end.

REPORT MEMORY=MVS

The MVS Memory Report provides a listing of the information about CA Datacom common memory, including ECSA, ESQA, and any other. It includes modules loaded as an SVC or sub-system or cross memory.

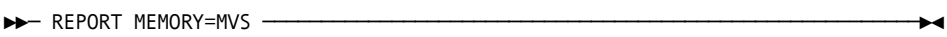
Reports generated using REPORT MEMORY=MVS provide information about all the memory used in the common memory pools.

When to Use

You can generate this report at any time.

How to Use

Following is the control statement to request this report:



Command

REPORT

Invokes the function to produce a report.

Required Keywords

MEMORY=MVS

Requests the Memory Report.

Note: An optional keyword DUMPS= is also supported, but *only* used at the direction of CA Support. The output from using DUMPS= is the normal report and additional SNAPS of memory. No protected memory is included in the report. Control blocks are for use by CA Support only and are therefore not described in detail externally.

If so directed by CA Support, the following values can be specified:

DUMPS=REQUEST

Includes some Operating System control blocks.

DUMPS=TRACE

Includes DUMPS=REQUEST plus and some CA Datacom system control blocks.

DUMPS=FULL

Includes DUMPS=TRACE plus CA Datacom SVCs, in some cases.

DUMPS=FULLX

Includes DUMPS=FULL plus additional control blocks. Only recognized SVCs are reported.

DUMPS=SECURITY

Includes no additional information.

Example JCL

The following shows the command to generate the Memory 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command Input
        REPORT   MEMORY=MVS
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                CONTROL CARD(S)
                .........1.........2.........3.........4.........5.........6.........7.........8
                REPORT  MEMORY=MVS

                FUNCTION=REPORT
                   MEMORY=MVS
```

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.

```
OPERATING SYSTEM z/OS   V1.9   PROCESSOR 2097   SYSTEM CA11
CA DATACOM/DB SUBSYSTEM IS PRESENT
```

This page of the report shows the following:

OPERATING SYSTEM

This line starts with OPERATING SYSTEM, followed by the name that CA Datacom has identified as the operating system (z/OS in the example).

Following the operating system is the version and release that CA Datacom has identified.

Following the version and release is the word PROCESSOR followed by a number (2097 in the example) that CA Datacom has identified as the processor.

Following the number of the processor is the word SYSTEM, followed by the name identified as the system name where the DBUTLTY function executed (CA11 in the example).

SUBSYSTEM IS PRESENT

If the SUBSYSTEM IS PRESENT message appears on the new line, it shows that the CA Datacom/DB subsystem is in the operating system. When the subsystem is not in the operating system, this line shows CA DATACOM/DB SUBSYSTEM IS ABSENT. This SUBSYSTEM is used by and supports older CA Datacom releases prior to Version 14.0. It is not needed or used by 14.0 and should not be installed if there are no older CA Datacom releases present.

```
COMMON MEMORY USAGE  (NOT SVC OR MUF SPECIFIC)
 ADDRESS      SIZE  TEXT
2F8AB000      4096  DB_SVC_AC R14.0+ SYSTEM ANCHOR
35FB4080  3AB51000      3540  DBPCCPR  14.0   20100727 15.55  CROSS MEMORY  PC=PROD
3AB51000      3540  DBPCCPR  14.0   20100727 15.55  CROSS MEMORY  PC=TEST
39448000      2738  DBPCSPR  14.0   20100808 10.36  PC XCF + RRS  PC=PROD
00C77D04        36  SSCT 'DB  ' AND SMALL ANCHOR
35FB4080       384  DBNSDBNB LARGE ANCHOR
35E65000      2082  DBESPPR  12         CROSS MEMORY
00C56000       632  DBSSVPR  12         SUBSYSTEM

NUM I TYPE OPT. LOCK  ADDRESS   DBSVC   DATE   TM/SP      SIZE
235    3/4           00B75D78   11.0  12/13/03 SP04       9932
239    3/4           394D0000   12    04/15/11 16.09      9024

MUF NAME  STATUS  REL  JOB NAME JOB ID   CXX NAME DC COMM31 ENBL-MDHMS COMMENTS
DBDVM01  ENABLED  14.0 DBDVM0   JOB58297 DBDVM0   DB   4736 0628103948

DBDVM02  ENABLED  12   DBDVM02  JOB58894 DBDVM02  DB   4928 0628105229 SVC=239
DLV1M11D ENABLED  11.0 DLV1N11D          AX11AD   AD1839360 0627221559 SVC=231 COMMON24=192
JLM12RRM ENABLED  12   DBDVM0RM JOB58895               4416 0628105254 SVC=239 RRMNAME=JLM12RRM
MSMDEVB  ENABLED  11.0 MF2MUFB  00010MUF CAMSM    AD  11904 0628124727 SVC=231 MSM COMMON24=192
MUFM     DISABLED 14   DSNZMUFM STC59537
QAMUFM   DB  23616            SVC=213                       MUFM1    DISABLED 12   DSNZMUFM STC59622
```

This page of the report shows the following:

COMMON MEMORY USAGE (NOT SVC OR MUF SPECIFIC)

This section of the report provides common memory that is not an SVC and is not part of a specific MUF. This section of the report can be empty if no common memory has been allocated. Each item is printed with its address, size, and text that gives identifying information. In the sample, note the following:

- The first line exists if Version 14.0 or a later release has built a system anchor.
- The next lines exist for the executing PC supporting modules in memory. Possibly 4 for Version 14.0 with one for the DBPCCPR module to be used for production and one for test and one for the DBPCSPR module to be used for production and one for test.
- The next lines exist is Version 12.0 or a earlier release has built a small and large anchor.
- The next lines exist for Version 12.0 or earlier if additional modules are in common memory supporting the Subsystem and cross memory.

The next section reports any Datacom SVC's that are installed.

NUM

NUM is the SVC number being reported.

I

I is the column that can contain the letter 'I' if the SVC appears to be invalidly defined.

TYPE

TYPE is the type of SVC such as 3 or 4.

OPT

OPT is the SVC option. Only blank or A are valid options for the CA Datacom/DB SVC.

blank

A blank indicates no option was specified.

A

An A indicates Assistable.

E

E indicates-use of the extended SVC router.

I

I indicates not preemptable.

P

P indicates the authorized program facility.

R

R indicates Access/Register mode.

LOCK

LOCK is the operating system lock required on entry to the SVC routine. The lock must be blank for the CA Datacom/DB SVC.

blank

blank indicates none required.

C

C indicates the CMS lock.

D

D - The dispatcher lock.

L

L indicates the local lock.

O

O indicates the optional lock.

S

S indicates the storage allocation lock.

ADDRESS

ADDRESS is the address of the location of the SVC.

DBSVC

DBSVC is the version number for CA Datacom/DB when this SVC was generated.

DATE

DATE is the date of the version for CA Datacom/DB when this SVC was generated.

TM/SP

TM/SP is a time of the version for the SVC.

Only SVC table entries from 200-255 are searched, and only those that appear as CA Datacom/DB are printed, except when the optional keyword SECURITY=YES is specified, in which case all SVC table entries in the range are printed (this assists in debugging, should that be necessary).

MUF Specific Common Memory Section:

MUFNAME

The MUF name, if known, is provided.

STATUS

The MUF status is provided as: ENABLED, DISABLED, EOJ SET.

Note: A shadow is not yet a MUF and has no identity.

REL

The Version release of the MUF last enabled

JOB NAME

The Job or started task name of the MUF last enabled

JOB ID

The Job ID or started task ID of the MUF last enabled

CXX NAME

The CXX name of the MUF last enabled

DC

The CA Datacom CXX type of DB for CA Datacom/DB, AD for [set the dad variable for your book], or blanks either if unknown or if this is an RRM address space supporting RRS.

COMM31

The size of common MUF specific memory allocated for this specific MUF.

ENBL-MDHMS

The date and time when the MUF was last enabled in the form of month-day-hour-minute-second

COMMENTS section can contain the following:

SVC=*nnn*

SVC=*nnn* occurs if the MUF is using a CA Datacom SVC.

SM

MSM occurs if this MUF is recognized as DATACOM MSM.

ENF

ENF occurs if this MUF is recognized as ENF, that is, if CA ENF is using a MUF within its own address space.

TR

TR occurs if this MUF is recognized as a Datacom TR

PC=TEST

PC=TEST occurs if this MUF is using the test versions of DBPCSPR and DBPCCPR

RRMNAME=*name*

RRMNAME=*name* occurs if this is not a MUF but an RRM supporting RRS

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.

With Simplify mode, the report is designed and intended to execute with the MUF enabled. If it is needed to execute with the MUF not enabled, it needs to follow a function noting MUF is not enabled by executing with SET OPTION1=MUF_NOT_ENABLED. Without Simplify mode, you can request a Statistics and Diagnostics Area report at any time.

Length of Time to Execute REPORT

Note: Each Statistics and Diagnostics Area (PXX) Report option that you request causes the Statistics and Diagnostics Area to be read from track 1 to the end of the current data or the end of the extents. If your Statistics and Diagnostics Area data is allocated in a large area, the elapsed time to execute this function can be lengthy.

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 MUF 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:

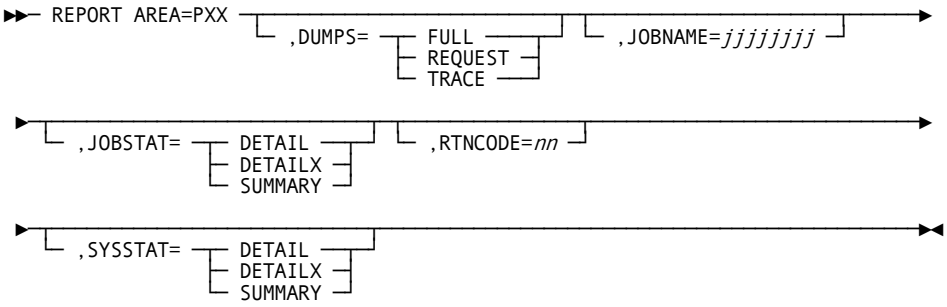
1. DBUTLTY REPORT AREA=PXX,SYSSTAT=SUMMARY gives SYSTEM summary STATS but not JOBSTAT summaries.
2. DBUTLTY REPORT AREA=PXX,SYSSTAT=DETAIL gives one copy of SYSTEM summary STATS with a jobname matching the MUF, plus JOBSTAT summaries for the same jobname, then the SYSTEM summary STATS again, then JOBSTAT summaries again. In this case, statistics are unnecessarily duplicated.

Note: For detailed information about MUF 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) for details on the COMM ALTER function.

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 memory management methods that support 64-bit addressing and are faster. Because of this enhanced processing and to simplify it, the following changes were made:

Dump Addresses Format Changes

The formatting of dumps has changed 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 MUF startup option.

STATBFR Startup Option Removed

You are not allowed to specify STATBFR in the MUF startup. The STATBFR MUF startup option was previously used to specify the size of the buffer that accumulates system and job statistics, but STATBFR was removed. Complete statistics are now always provided when requested.

SNAPER Processing Changes

In z/OS environments, output SNAPER processing has changed. Previously, full address space SNAP dumps and partial memory were all written to SNAPER. An address space SNAP, produced due to an abend, is routed to SNAPER the same as it was. All other items, however, are written to other SYSOUT data sets to keep them separate and make them easier to find.

Another change is that in a MUF failure, any ECSA memory is written to a DD statement, SNAPECSA, and data space memory, when used for communication, is written to SNAPDTSP. During debugging, if multiple address space dumps are snapped, they are individually written to SNAPAS*nn* where *nn* is a number 01 through 99.

ECSA Labels

The Master List in the ECSA is labeled DB_ML_ECSA.

Message DB02406I Changed

To support the change in dump processing, 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 that have been removed. The process now has a single completion. In addition, text describing the error has been added to the end of the message.

Note: 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 765) and COMM ALTER (Modify Error Table) (see page 196).

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 MUF startup option or specify it as PXXSTATS DETAIL. Detailed information about MUF 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 MUF startup option. For detailed information about MUF 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)

SYSOUT Processing

If the SYSOUT MUF startup option specifies using PXXML, there are no DUMPS in the Statistics and Diagnostics Area(PXX) data set. This means that running a PXX report would run without errors but find nothing.

Note: For more information on SYSOUT processing, see the CA Datacom/DB Database and System Administration Guide.

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.

```
//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
//PXX      DD DSN=pxx.data.set,DISP=SHR          PXX data set
//SYSIN    DD *                                  Command Input
         REPORT   AREA=PXX,JOBSTAT=DETAIL,SYSSTAT=SUMMARY
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  REPORT  AREA=PXX,JOBSTAT=DETAIL,SYSSTAT=SUMMARY

                  FUNCTION=REPORT
                     AREA=PXX
                     JOBSTAT=DETAIL
                     SYSSTAT=SUMMARY


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
```

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.

  The first message in the example just shown notes that the PXX was started when the MUF was enabled. 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 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. In z/VSE 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:

```
PXX RESET BY CLRPXX                        JOB=DBDVM0              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
```

```
DATA BASE  STARTED AFTER COMM CLRPXX       JOB=DBDVMUF           SUBID=0                1/06/2003 7.10.15
```

This page of the report can contain information such as the following:

- A message indicating the start of the MUF. It includes the job name, the sub-identifier, the date, and the time.
- If you have CA Datacom DB2 Transparency installed, the message SQL INITIALIZATION: DB2/T PRESENT is printed.

```
DATA BASE  JOB STATISTICS              CLOSING    JOB-81471-DRH402E   TASK-001            1/06/2003 16.24.01


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
```

This page of the report displays statistics for each job that wrote statistics as follows:

Database Job Statistics

DATA BASE JOB STATISTICS

A message, such as the example, begins the job statistics section. 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.

TABLE

Displays the DATACOM-NAME of the table concatenated with the DATACOM-ID of the database containing the table.

URT

Displays the number used by CA Datacom/DB to keep track of the User Requirements Table. For SQL, this number is always zero.

SQBUF

Displays the number of sequential buffers.

UPD

Displays 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.

SYN

Displays 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.

SUB

Displays the value for DBFLSUB= in the User Requirements Table. YES indicates the CA Datacom/DB table load subroutine DBFLSUB is being used.

DXC

Displays 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.

GBL

Displays the User Requirements Table parameter GETBLK= is being used to specify the buffer size used for blocked GETIT and GETPS commands.

REQUESTS

Displays 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.

RETRIEVALS

Displays 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.

```
DATA BASE  JOB STATISTICS              CLOSING    JOB-81471-DRH402E   TASK-001            1/06/2003 16.24.53


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
     .    .     .    .   .   .   .   .           .          .          .           .           .
     .    .     .    .   .   .   .   .           .          .          .           .           .
     .    .     .    .   .   .   .   .           .          .          .           .           .
     .    .     .    .   .   .   .   .           .          .          .           .           .
```

This page of the report displays the following information:

ADDS

Displays the number of add requests recorded against this table. CA Datacom/DB commands and SQL statements are included in this count.

DELETES

Displays the number of delete requests recorded against this table. CA Datacom/DB commands and SQL statements are included in this count.

UPDATES

Displays the number of update requests recorded against this table. CA Datacom/DB commands and SQL statements are included in this count.

```
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
LOG AREA MAXIMUM PERCENT FULL              13
SECONDARY E/C CONFLICTS                     0
TASK COMPLETION WAITS                       0
REQUEST COMPLETION WAITS                    0
SPILL COMPLETION WAITS                      0
```

This page of the report displays the following information:

Data Base Summary Statistics

REQUESTS

Displays 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 MUF, not the number of detail requests for the next record.

SQL REQUESTS

Displays the number of requests to the MUF 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 MUF.

DATA MANAGER REQUESTS

Displays the number of calls to the low-level CA Datacom/DB processor. Each CA Datacom/DB command issued to the MUF 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.

I/O EVENTS READS/WRITES

Displays 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.

REQUESTS

Displays the number of times primary exclusive control was granted for any data record. This field reports exclusive control activity.

CONFLICTS

Displays the number of times a task requested primary exclusive control of a record that was currently held under primary exclusive control by another task. This field reports exclusive control activity.

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.

K/E BUFFER REORGANIZATION

Displays the number of times a key/element definition had to be removed from memory to allow a different key/element definition to be read.

NO ... BUFFER AVAILABLE

Displays 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.

LAST ...

Displays the total number of references to a particular class of buffers.

... BUFFER USED

The counters tell you how much the IXX, DXX, and data buffers allocated in the MUF 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.

SEQ. READ AHEAD I/O EVENTS

Displays 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).

LOG AREA MAXIMUM PERCENT FULL

Displays the highest percent full the Log Area reached.

SECONDARY E/C CONFLICTS

Displays 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.

... COMPLETION WAITS

Displays the number of times requests had to wait for service. Continues on the next page of the report.

TASK COMPLETION WAITS

Displays 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

Displays 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

Displays 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.

```
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
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
```

This page of the report displays the following information:

Data Base (Continued)

... COMPLETION WAITS

Displays the number of times requests had to wait for service.

ACCOUNTING COMPLETION WAITS

Displays 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 MUF startup options.

SECURITY COMPLETION WAITS

Displays the number of times that security had to wait for system information.

INDEX QUEUE PROCESSING

Displays the number of index queue entries processed. This field reports index queue. Queue processing includes:

- All high-level block processing
- Block deletions of low-level blocks

This statistic is useful primarily to CA Support for debugging purposes.

INDEX QUEUE OVERFLOWS

This field is not currently used and therefore contains a zero.

INDEX BLOCK SPLITS

Displays 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 CA Support for debugging purposes.

CBSBFR SPILL COUNT

Displays 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 MUF startup option CBS.

CBSBFR ENTRIES AVAILABLE

Displays the number of 185-byte entries available in the Compound Boolean Selection buffer (defined by the MUF startup option CBS value) for storing active set definitions. A set definition typically requires one to three entries.

MOST CBS ENTRIES USED

Displays 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

Displays the total number of sets processed.

SETS REQUIRING TEMPORARY INDEX

Displays the number of sets processed that built a temporary index.

TEMPORARY INDEX ENTRIES BUILT

Displays the number of temporary index entries built to satisfy SELFR command processing.

CBS MEMORY ALLOCATIONS

Displays 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.

WRITEPEND WRITE IO

Displays the number of logging (physical) writes (controlled by the MUF startup option LOGPEND) that occurred during this cycle.

DATASECURE…

These fields are reserved for future use.

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

Displays 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 MUF 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

Displays 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

Displays the number of times that a log buffer was written because it was full.

LOG COMMAND WRITE I/O

Displays 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, and 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 MUF 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

Displays 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

Displays the number of write I/O to the LOGPOOL buffers used by transaction backout.

LOG 2-PHASE COMMIT WRITE I/0

Displays 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+

Displays the secondary data area buffers allocated in the MUF 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

Displays 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.

DATA BASE SUMMARY STATISTICS 3/05/2014 23.17.40

TABLE/AREA	REQUESTS	I/O EVENTS - READS	I/O EVENTS - WRITES	LOGICAL READS	LOGICAL WRITES	
CXX	0	59	6	256	17	A
FXX	0	2	1	2	0	A
LXX	0	3	13	3	351	A
IXX00006	0	0	0	4	0	A
A0100297	0	7	2	362	702	A +
C0100297	20	0	0	0	0	T+
C0200297	30	0	0	0	0	T+
F0100297	40	0	0	0	0	T+
F0200297	50	0	0	0	0	T+
F0300297	60	0	0	0	0	T+
IXX00297	0	4	2	167	81	A +
I0100297	0	4	2	445	221	A +
I0200297	0	4	2	365	181	A +
I0300297	0	4	2	265	131	A +
I0400297	0	4	2	265	131	A +
I0500297	0	4	2	145	71	A +
I0600297	0	4	2	145	71	A +
I0700297	0	4	2	145	71	A +
I0800297	0	4	2	245	121	A +
I0900297	0	5	5	149	73	A
K0100297	70	0	0	0	0	T
RNA00297	80	0	0	0	0	T
RNB00297	0	0	0	0	0	T
VRA00297	0	0	0	0	0	T
VRB00297	0	0	0	0	0	T
TOTAL	350	112	45	2,963	2,222	

This page of the report displays columns that 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.

REQUESTS

Displays the number of requests per table which have occurred and been recorded in the Statistics and Diagnostics Area.

I/O EVENTS - READS

Displays the number of I/O reads which have occurred and been recorded in the Statistics and Diagnostics Area.

I/O EVENTS - WRITES

Displays the number of I/O writes which have occurred and been recorded in the Statistics and Diagnostics Area.

LOGICAL READS

Displays the number of internal read requests which have occurred and been recorded in the Statistics and Diagnostics Area.

LOGICAL WRITES

Displays the number of internal write requests which have occurred and been recorded in the Statistics and Diagnostics Area.

Following the LOGICAL WRITES column

Following the LOGICAL WRITES column, there may be a letter that indicates more information as follows:

A

indicates that the statistics line pertains to an AREA.

T

indicates that the statistics line pertains to a TABLE.

Following the “A” or “T” there may be a “+” (plus sign)or a “-“ (minus sign).

+

indicates that the table or area on that line was successfully processed (and possibly updated) during ESTAE processing if the MUF abnormally. terminated.

-

indicates that the table or area on the line was examined during the ESTAE processing but did not complete successfully.

```
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
```

This page of the report displays the following information:

Database Summary Requests Totals

RETRIEVALS

The total number of retrieval requests since MUF 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

Displays the total number of add requests since MUF enable.

DELETES

Displays the total number of delete requests since MUF enable.

UPDATES

Displays the total number of update requests since MUF 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 777)). 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 CA 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 definition

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 Datadictionary 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//PXX      DD DSN=pxx.data.set,DISP=SHR          PXX data set
//SYSIN    DD *
        REPORT   AREA=PXX,DUMPS=TRACE
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Sample Report: REPORT PXX - CBS Facility Diagnostic Report – Example 1

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT AREA=PXX,DUMPS=TRACE

FUNCTION=REPORT
   AREA=PXX
   DUMPS=TRACE
```

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.

```
*** 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
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***
```

This page of the report displays the following information:

Message at top of page

Request is made by CA Ideal program CBSTS001, version 001, system $ID, on line 000102.

REQUEST SPECIFICATION section

The 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.

OPTIMIZATION section

The 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.

SELECTION section

The 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.

ORDERING section

The 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*.

RETRIEVAL section

The 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.

Sample Report: REPORT PXX - CBS Facility Diagnostic Report – Example 2

```
 ** 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
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***
```

This page of the report shows the following:

Message at top of report

The request is made by CA Ideal program CBSTS005, Version 001, system $ID, on line 000102.

REQUEST SPECIFICATION section

The section displays the following information:

DATAVIEW ELEMENT:

Displays 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:

Displays the 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:

Displays 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:

Displays the 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.

OPTIMIZATION section

The 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.

SELECTION section

The section displays the following information:

TYPE SELECTION:

Selection is DIRECT.

ORDERING section

The section contains the following information:

TEMPORARY INDEX REQUIRED DUE TO:

The ordering criteria, because a signed numeric field is specified.

ENTRIES INDEXED:

Displays that 200 rows meet the selection criteria.

LOGIO:

This requires 609 index and data buffer references to build the temporary index.

RETRIEVAL section

The 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 ***


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
```

This portion of the report displays the following information:

GRP

GRP - "OR" group number. An "OR" group consists of predicates joined by AND. Statistical population counting is performed separately for each "OR" group.

LVL

Displays the 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.

KNAME

Displays the 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.

CNTKR-B

Displays the "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.

LOW EST/HIGH EST

Displays the 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.

(TINDX)

Displays the 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.

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 CA 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;
```

Note: 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.

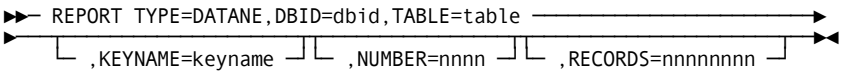
Security Setting: The DBUTLTY external resource name for the DATANE report is REPORT.DATANE.

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       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.

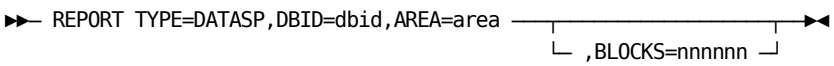
Security Considerations: The DBUTLTY external resource name for the DATASP report is REPORT.DATASP.

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.

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: History does not include physical backups or EXTRACTs without UPDATE=YES (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 MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

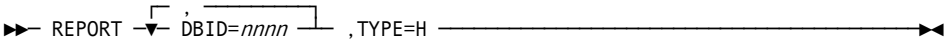
When to Use

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 MUF enable were executed.

This function executes through the MUF and, therefore, the MUF must be enabled. All tables must be loaded with a valid index because each area is opened with read status.

How to Use

The MUF must be active when you 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.

```
//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   DBID=1,TYPE=H
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
REPORT DBID=1,TYPE=H

FUNCTION=REPORT
   DBID=1
   TYPE=H
```

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.

```
                  FUNCTION HISTORY REPORT


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
       BACKUP (-2)
       BACKUP (-3)
       BACKUP (-4)
       BACKUP (-5)
       BACKUP (-6)
       BACKUP (-7)
```

This page of the report shows the following:

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.

EVENT

A list of the functions that have executed against the area.

JOB NAME

The name of the job that executed the function.

DATE

he date the function was executed.

TIME

The time the function was executed.

CXXNAME

The name of the Directory at the time the function executed. The CXXNAME is only available for the INIT and the last LOAD.

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.

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).

TYPE=U - Debugging Report

(z/OS only) The DBUTLTY function, REPORT TYPE=U, identifies selected CA Datacom/DB data sets for debugging purposes.

When to Use

Users occasionally encounter a situation in which CA Datacom/DB rejects a data set. If this happens, information about what the rejected data set contains might not be apparent.

DBUTLTY TYPE=U debugging reports can identify *selected* data sets that CA Datacom/DB can recognize. TYPE=U reports are not intended to identify all data sets from all releases. Because TYPE=U debugging reports are ways to only *identify* data sets, if details about a CA Datacom data set are needed, detailed reporting tools can be used, for example, DBUTLTY functions REPORT AREA=CXX or REPORT AREA=IXX.

Note: To minimize confusion, do not give DDNAMEs that you think the data set is, such as CXX, LXX, IXX002, and so on.

Security Considerations

Like most DBUTLTY functions, the REPORT function is subject to external security. If the MUF identified by the working DBSIDPR or alternate (as generated by the DBSYSID macro) is enabled and available and secured, the REPORT gets execution rights. If the MUF is not enabled or is not secured, the function continues. If the CXXNAME from the MUF is available, it is provided in the report heading.

Note: For DBUTLTY External Security, REPORT TYPE=U,DDNAME uses REPORT.DDNAME with no table rights.

How to Use

A TYPE=U report function opens a data set, reads the first block, and (if possible) identifies it from the following example list of data sets subject to identification by pointing to a DD statement with a data set name that contains unknown data needing identification. Therefore, to use a TYPE=U report function, the DDNAME must be provided, the data set must be able to be opened, and the first block of the data set must be able to be read.

You must have read access to a data set to execute a TYPE=U report function, because the data set to be identified is opened for read processing in the DBUTLTY address space.

Execute the TYPE=U report function with the following syntax.

DD-name

The *DD-name* specifies the JCL statement needing identification.

The TYPE=U report function does not open any data set not named in the DDNAME= keyword, and no data set opened is fully validated as the data set it appears to be. Identification of a data set does not necessarily make it the specific data set you wanted to identify. CA Datacom does not uniquely own the byte values used for identifying data sets, and misidentification is therefore possible. Also note that CA Datacom does not currently, nor does CA Datacom have any plans to ever provide field layouts of any CA Datacom data set.

A DDNAME is not acceptable for sequential input or output files if it is a name reserved for a CA Datacom area. Names with the following patterns are therefore not acceptable for DDNAME=:

- 3-byte names that end with XX, meaning they are reserved as either current or future CA Datacom control areas.
- 6-byte names that end with what could be a database ID from 001 through 999.
- 7-byte names that end with what could be a database ID from 1000 through 9999.

The DDNAME= value is verified for acceptability to protect you from unintentionally causing data corruption. The DDNAME check is the default but optional. You can prevent the DDNAME check by using a DBSIDPR parameter (DBUTLTY_EDIT_DATA_SET=) for individual MUF environments. However, we recommend that you allow the DDNAME check.

The data corruption risk involves not the DDNAME itself but the content of the data set. For example, suppose that you used the CXX DDNAME as the output of a backup. You then copied the CXX DD statement and changed the DDNAME of the copy to be acceptable, avoiding the DDNAME= error. The backup would, however, then overlay the CXX data set, which is not the intent of a backup.

If you specify an unacceptable name for DDNAME=, message DB10059E is generated. For more information about DB10059, see the *CA Datacom/DB Message Reference Guide.*

Note: We recommend that you allow DDNAME= check protection. You can, however, disable DDNAME= protection. To disable protection, assemble the DBSIDPR module used for this CA Datacom environment and specify NONE for the DBUTLTY_EDIT_DATA_SET= parameter. The default is DBUTLTY_EDIT_DATA_SET=FULL_1, which allows DDNAME= protection. For more information about DBSIDPR and DBUTLTY_EDIT_DATA_SET=, see the *CA Datacom/DB Database and System Administrator Guide.*

Examples

The TYPE=U report function prints a heading and either a line of identifying information or a message stating that the data set is not recognized. Not every data set is subject to being identified. Data sets that can be identified are listed in a following example. Notice specifically that because an EXTRACT file has no control information, it cannot be identified. Whether or not the data set is identified, the block is snap dumped. The snap dump is needed if CA Support is required to further identify the contents.

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Example 1

This example shows what happens when a data set cannot be identified.

In the following example report, the specified data set with a DDNAME of DSX1 is not known to identification.

Note: If no block can be successfully read, no block is printed.

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  REPORT  TYPE=U,DDNAME=DSX1

                  FUNCTION=REPORT
                     TYPE=U
                       DDNAME=DSX1
```

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.

```
DATA SET IDENTITY INFORMATION                                                  DEVICE   INIT DATE TIME  CXX NAME
CANNOT IDENTIFY DATA SET
000000  A1F4B940 D3C9C2D9 C6F0F207 40404040   0384C2D3 D2E2E9C5 1D7C4040 40404040     *   *
```

Further information follows the report, as needed, to snap dump the block.

Example 2

In this example, the data set is identified as a DIRECTORY CXX having a CXX LEVEL of 01 and being CA Datacom/DB (indicated as DB in the report). The DB would be replaced with AD if this CXX was built for CA Datacom/AD. In this example, the device type, 3390, is the device the content of the CXX indicates is the device type. It is required to match the current external device type for normal CA Datacom processing. The INIT DATE TIME is 2010 January 17 at 10.39.53 am and the CXX name is DBDVMUF.

```
DATA SET IDENTITY INFORMATION                                          DEVICE   INIT DATE TIME  CXX NAME
DIRECTORY CXX R12 LEVEL 01 DB                                            3390  20100117 083953  DBDVMUF
```

A block 1 dump, not shown in this example, occurs next.

Example 3

Example 3 is a CXX for r11. A CXX for r10 or r9 are also noted, but older than r9 would be shown as release _?.

```
DATA SET IDENTITY INFORMATION                                          DEVICE   INIT DATE TIME  CXX NAME
DIRECTORY CXX R11 DB                                                     3390  20091115 081827  QAMUFZ
```

A block 1 dump occurs next but is not shown in this example.

Example 4

This example involves a CXX backup for r11. Notice that a CXX backup does not know the device or INIT date/time of the CXX at the time of the backup. A backup for a CXX would print similar information.

```
DATA SET IDENTITY INFORMATION                                          DEVICE   INIT DATE TIME  CXX NAME
BACKUP DIRECTORY CXX R11 DB                                                                    DB11MUF2
```

A block 1 dump occurs next but is not shown in this example.

Example 5

This example involves a LXX with information about whether it is variable, fixed not spanned, or fixed spanned. Notice that the LXX does not provide a device type or CXX name, and it only provides a date/time of INIT if it was initialized with r11 SP2 or later.

```
DATA SET IDENTITY INFORMATION                                          DEVICE   INIT DATE TIME  CXX NAME
LOG LXX VARIABLE                                                              20080511 170628
```

A block 1 dump occurs next but is not shown in this example.

Example 6

This example involves a Force Area (FXX). Notice that the FXX does not provide a device type or CXX name and only provides a date/time of INIT if it was initialized with r11 SP2 or later.

```
DATA SET IDENTITY INFORMATION                                                DEVICE   INIT DATE TIME  CXX NAME
FORCE FXX                                                                           20080512 061600
```

A block 1 dump occurs next but is not shown in this example.

Example 7

This example involves an IXX (Index Area) index. Historically, CA Datacom/DB supported an index from r7.3, but the INIT date/time was not reported until r8, and the century was not reported until r10.

If the index were not an IXX but a valid multiple data set index, the identification would show a 3-character name such as I01 (I zero one) instead of the IXX shown in this example.

```
DATA SET IDENTITY INFORMATION                                                DEVICE   INIT DATE TIME  CXX NAME
INDEX IXX 0997                                                                 3390  20080607 120619  DBDVMUF
```

A block 1 dump occurs next but is not shown in this example.

Example 8

This example involves a data area. Fully supported data areas are those initialized since CA Datacom/DB r8.0. Data areas initialized by CA Datacom/DB r7.4 or r7.5 and opened at least once for update by r8.0 through r10 (and which were converted to r8.0 style) are reported, but the INIT DATE TIME fields might not have been properly set, and the century might or might not be valid.

```
DATA SET IDENTITY INFORMATION                                                DEVICE   INIT DATE TIME  CXX NAME
DATA AREA  A01 0997                                                            3390  20100530 070514  DBDVMUF
```

A block 1 dump occurs next but is not shown in this example. If this data area contains tables defined with encryption, the key handle in use is provided and the table information. For example:

```
ENCRYPTION INFORMATION - BASIC
KEY 1 - 53F61DC40FDC626E6E365F8221035DAD6D35DEBF36CFC8513DC1644445E95D55
TABLE ID CMPRS RCV  DATA USER COMPRESSON   T M
C01    1 NO    YES 32720                   B A
C02    2 DB    YES 32720                   B B
F01  103 NO    YES 32720                   B C
VRA   98 NO    YES    84
```

Example 9

This example involves a work area (WXX) for RRS two-phase commit processing. Notice that the WXX has no device, INIT date/time, or CXX name information.

```
DATA SET IDENTITY INFORMATION                                          DEVICE   INIT DATE TIME  CXX NAME
RRS WXX
```

A block 1 dump occurs next but is not shown in this example.

Example 10

This example involves a MUF temporary print area PXX (Statistics and Diagnostics Area). Notice that the PXX has no device, INIT date/time, or CXX name information.

```
DATA SET IDENTITY INFORMATION                                          DEVICE   INIT DATE TIME  CXX NAME
PXX
```

A block 1 dump occurs next but is not shown in this example.

Example 11

This example involves a data area backup produced by the DBUTLTY BACKUP function from releases starting with CA Datacom/DB r10. Before r10, there was no identification information available. Unknown fields not printed are DEVICE and INIT DATE TIME. The area name is blanks if the BACKUP function did not include the AREA= keyword.

```
DATA SET IDENTITY INFORMATION                                          DEVICE   INIT DATE TIME  CXX NAME
BACKUP DATA AREA  A01 0997                                                                     DBDVMUF
```

A block 1 dump occurs next but is not shown in this example.

Chapter 37: 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 MUF 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 Simplify mode, the RESET AREA=CXX is not needed and not allowed to execute. Without Simplify mode, in processing a reset for the CXX DBUTLTY attempts to connect to the MUF. If successful, the MUF 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 MUF is not enabled or cannot be checked for any reason, the reset continues.

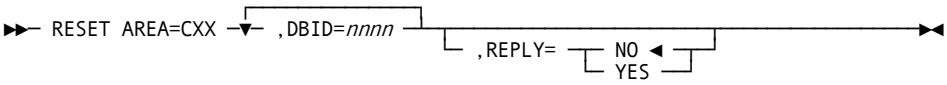
This function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

How to Use

You can execute this command in either Single User or with the MUF active. 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.

```
//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
        RESET     AREA=CXX,DBID=10
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    RESET AREA=CXX,DBID=1

                    FUNCTION=RESET
                       AREA=CXX
                       DBID=00001
```

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.

```
                    CONTROL FILE (CXX) OUTPUT --------------------------- EXCLUSIVE CONTROL NOT HELD
                                               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
```

This page of the RESET AREA=CXX report shows the OPEN status of databases as YES or NO as follows:

STATUS INDICATORS

Information provided by the STATUS INDICATORS.

Message at the end of report

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. It should not be executed on a test system either, or any system. It can leave the system databases such as CA Datacom Datadictionary bases 2 and 15 out of sync with themselves or each other.

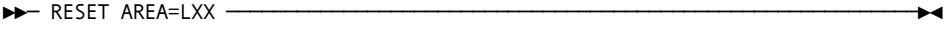
Do not use this function if MUF is enabled. In Simplify mode, edits prevent execution with MUF enabled. In Simplify mode, the function may be the first function that requires the CXX and protects it as MUF not enabled, or it may follow another function that provides the protection. In Simplify mode, this must be the only function with AREA=LXX allowed in the input stream. It may not occur either before or after an INIT of the LXX or a REPORT of the LXX. If this function is being considered, review the MUF startup options DORESTART and NORESTART to see if they would provide resolution without resetting the LXX.

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 591).

How to Use

You can execute this command in either Single User or with the MUF active. 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.

```
//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
//LXX      DD DSN=lxx.data.set,DISP=SHR          Log area data set
//SYSIN    DD *                                  Command Input
         RESET     AREA=LXX
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
RESET AREA=LXX

FUNCTION=RESET
   AREA=LXX
```

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.

```
                                        LOG FILE (LXX) OUTPUT


DB13084I - LOG AREA RESET IS COMPLETE
```

This page of the report shows a message indicating the Log Area was successfully reset.

Chapter 38: RETIX (Reconstruct Index)

Use the RETIX function to rebuild:

- The data space index for one area or all areas of a database (see the following information).
- The data row count for one area or all areas of a database (see the following information).
- The Transaction Sequence Number (TSN) for one area or all areas of a database (see the following information).
- The Index Area (IXX) for one or all data areas of a database. (See Rebuilding an Index Area (see page 822).)

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 COMM STATUS (Print Status of MUF Jobs) 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, 3, 4, or 5). For an explanation of the CA Datacom/DB data area space management options, see the *CA Datacom/DB Database and System Administration Guide* .

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 821)).

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 821).

Any MUF 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.

MUF Connection when KEYNAME=*DATA or KEYNAME=*SETR

This function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

Using KEYNAME=*SETR to Recount Records and Update CXX

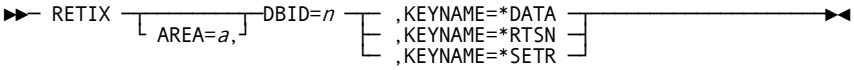
The count of records in a table, as known to the CXX, can become wrong if the MUF is restarted after a MUF outage occurs. The DBUTLTY RETIX function, with the KEYNAME=*SETR option, lets you recount the records and set the new value in MUF. Using the keywords DBID=, AREA=, and KEYNAME=*SETR with RETIX 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 and only if MUF is not currently running as a full MUFplex with two or more enabled or logically executing MUFs.

How to Use

When using the RETIX function to rebuild the data space index, the MUF must be active. This utility function executes only in the MUF. 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.

Required Keywords

,DBID=

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

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 about *DATA, see MUF Connection when KEYNAME=*DATA or KEYNAME=*SETR.

Use KEYNAME=*RTSN to detect and correct the high URI number. For more information about *RTSN, see Resetting the TSN (see page 837).

Use KEYNAME=*SETR to specify that records by recounted. For more information about *SETR, see Using KEYNAME=*SETR to Recount Records and Update CXX (see page 821).

Valid Entries:

*DATA, *RTSN, *SETR

Default Value:

(No default)

Optional Keywords

AREA=

Identifies the data area for which the data space index is to be rebuilt. This keyword can omitted to cause action for every area in the database.

Valid Entries:

DATACOM-NAME of the area

Default Value:

All areas in the database.

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) 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. With MINIMAL=NO (the default), RETIX reads the data area in physical sequence, extracted current key definitions, and merges them into the index in a process that also deleted any unnecessary index entries. With MINIMAL=YES, 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=YES 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=YES. 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 MINIMAL=YES 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 I2. 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 MUF 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 MUF.
- The database may *not* be open for update in another MUF 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 MUF if more than one is executing.

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

- The database will be opened by MUF for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets RETIX executing this MUF this database or area.

Successful Execution Requirements and Controls

Environmental **Requirements** *when MULTUSE=NO, SIMPLIFY NO, or MUF down*

- The database may *not* be open for update anywhere.
- The database may *not* be open in any MUF for read.
- The database may *not* have any table in *unloading* status.

Environmental **Requirements** *when MULTUSE=NO, Simplify YES and MUF enabled*

- The database may be open for update but must have no users.

Environmental **Controls** *when MULTUSE=NO, SIMPLIFY NO, or MUF down*

- 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 **Controls** *when MULTUSE=NO, Simplify YES and MUF enabled*

- The database will be opened for update if not already.
- All areas will be closed if open.
- The utility will execute with every ACCESS database or area status.
- The utility sets protection to block other incompatible DBUTLTY functions.

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

- The database may or may not be open for update in the connected MUF.
- The database may *not* be open for update in another MUF 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 MUF if more than one is executing.

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

- The database will be opened by MUF for update.
- The utility sets ACCESS area OFF (utility set).
- The utility sets RETIX executing this MUF this database or area.

Dynamic Extend during RETIX

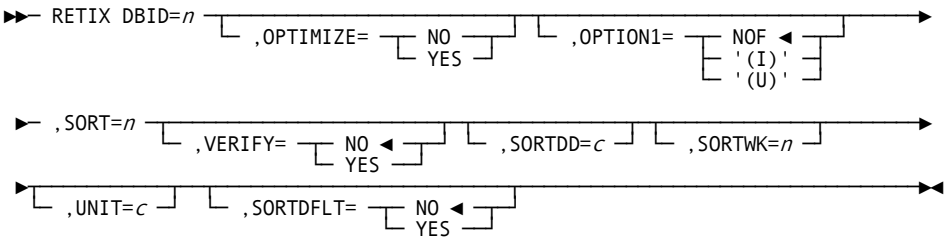
Dynamic extend of an index area is attempted if requirements are met. If requirements are not met, dynamic extend is not attempted. The requirements are as follows:

- SORT must be specified as greater than 0 (zero).
- The data areas must be using the URI option.

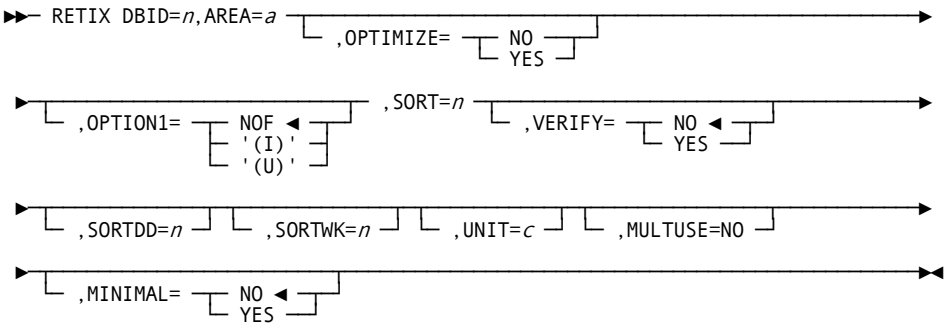
How to Use

You can execute this command in either Single User or with the MUF active. Execute the RETIX function using the following command format.

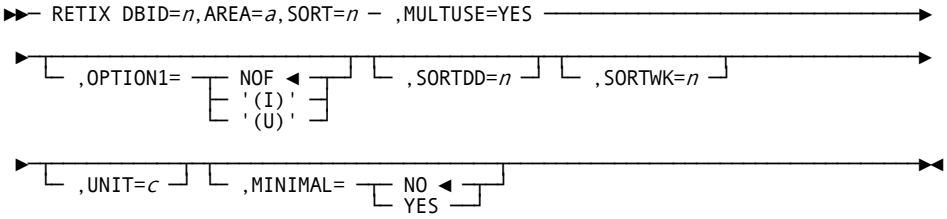
Rebuilding a Full Index Area



Rebuilding an Area's Index in Single User



Rebuild an Area's Index Area in MUF



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 previous releases. 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 MUF and Single User modes. The database involved must be able to be opened for update in the MUF 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 MUF 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 MUF. With MULTUSE=YES:

- The area is first opened in MUF 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 MUF address space. Packages of index entries (about 32k bytes) are sent from the utility to the MUF.

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 with the index and data areas data sets in the DBUTLTY address space.

For more information on area level DBUTLTY control, see Area Level DBUTLTY Control.

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/OS only) Specifies the statistics display option to be passed to the sort program (see the SORT parameter).

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. The DBUTLTY LOAD function also accepts any 3-character (including the surrounding parentheses) FLAG parameter that can be processed by the IBM sort. See your IBM sort documentation for details.

Valid Entries:

NOF, '(I)', or '(U)'

Default Value:

NOF

,SORT=

Activates the Index Area (IXX) entry sort option. When you specify a value, no IXX entries are created as the data is retrieved. The required IXX 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, and we recommend it 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.

This parameter's value specifies the estimated number of entries to be sorted. Sort treats each entry as a record. To determine the number of entries:

- Count one entry for each key definition for each row in the area.
- For any key defined with INCLUDE-NIL-KEY=NO, entries which would be low values or blanks are not passed to the sort.

- If the area is defined with space management option 1, 2, 4, or 5, entries exist for space management.
- If the area is defined with space management option 3, entries exist for both clustering and space management.

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. If a number is supplied to the sort, the Index Area entries in the Index Area are not deleted by a separate Index Area pass. The Index Area is updated and only the Index Area blocks that reflect change are written. This option is strongly recommended since it can improve performance. The OPTION1= parameter is also highly recommended to obtain warning and critical messages if a problem occurs.

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 that are also blocked during RETIX executions when the sort option is not being used are as follows:

- On-demand dynamic extend of any 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.

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)

SORTDD=

Specifies an optional DD statement, provided to the SORT that redirects SYSPRINT-type output from the SYSPRINT DD (used by DBUTLTY) to this optional DD statement.

Valid Entries:

printed output directed to optional DD statement

Default Value:

printed output directed to SYSPRINT, intermixed with DBUTLTY output

,SORTDFLT=

(z/OS only) Specify SORTDFLT=YES when you want the DBUTLTY SORT= value to be overridden with what is expected to be a more accurate number.

SORTDFLT= can be set to YES or NO (the default). If SORTDFLT=YES, the SORT= value is required to be greater than zero.

The keyword, SORTDFLT= allows the SORT value, when specified as greater than zero (0) for a RETIX to be overridden with an estimate expected to be more accurate. SORTDFLT= can only be used in conjunction with the SORT= keyword. When set as SORTDFLT=YES, the RETIX uses the current CXX count of records and definition to calculate a value for the SORT= override. If the count is zero (no records) for all tables to be loaded, the SORT=n value is used and not overwritten. Verify that it is in the appropriate range to allow a successful RETIX.

The record count and the number of key definitions per table are multiplied for each table to be loaded, and the products are summed as the estimate for record index entries. Based on the areas space management (DSOP) options, additional counts are added to verify a successful sort.

When using this DBUTLTY RETIX keyword, take into consideration that the number of records in a table after a MUF outage is likely to be inaccurate.

Valid Entries:

NO or YES

Default Value:

NO or YES

,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

,UNIT=

(z/OS only) Specifies the unit name for sort work areas. The parameter is always edited, but is only used when the SORT= parameter is specified as not zero.

Valid Entries:

1- to 8-character unit name

Default Value:

SYSDA

,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.

```
//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

//*
//* Note:  The following sort work allocations are not
//*        required if your sort package can dynamically
//*        allocate sort work areas.  Be sure to provide
//*        sufficient space for all keys to be sorted.
//SORTWK01 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SORTWK02 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//SORTWK03 DD UNIT=disk,SPACE=(alctyp,(prim,scnd))
//*
//SYSIN    DD *                                  Command Input
         RETIX     DBID=1,AREA=DEM,OPTIMIZE=YES,SORT=1000
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
RETIX  AREA=DEM,DBID=1,OPTIMIZE=YES,SORT=1000

FUNCTION=RETIX
   AREA=DEM
   DBID=1
   OPTIMIZE=YES
   SORT=00001000
```

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.

```
TABLE          RECORDS
PNC                 0
PNM                 0
POH                 0
POL                 4

TOTAL               4
```

This page of the report shows 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.

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:

```
TABLE          RECORDS

C01              6,000
TOTAL            6,000

TABLE       KEY NAME LOADED
C01              ANEWK
F01              ZNEWK

KEY ID DELETED PROCESSED
NONE
```

This page of the report shows the following:

KEY NAME LOADED

Displays 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.

KEY ID DELETED PROCESSED

Displays 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.

```
TABLE          RECORDS

C01                 9
TOTAL               9

TABLE       KEY NAME LOADED
C01              BNEWK
c01              CNEWK

KEY ID DELETED PROCESSED
     8
    10
```

This page of the report shows that in Single User (the default), all deleted keys are printed. In MUF, 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.

SORTDFLT=YES Report Section for RETIX

If SORTDFLT=YES, a SORTDFLT INFORMATION section occurs in the RETIX report before the table and count information.

The line after the SORTDFLT INFORMATION heading contains the following:

- Table name
- Record count from the current CXX
- Count of key definitions for the table
- Estimated count of keys to sort

The next line, starting with *DSOP CLUSTER, is the count of index entries possible for clustering (DSOP 3). The count is the same as the number of records, a ‘worst case’ estimate unlikely to occur because it presumes that every record has a unique cluster value, which would make clustering a poor choice for this data area.

The next line, starting with *DSOP PARTIAL, is the count of index entries possible for partial space (DSOP other than 0).The count is the number of data blocks divided by 56, a ‘worst case’ estimate unlikely to occur because it presumes that every block group of 56 blocks has at least one block with space for a record.

The next line, starting with *DSOP EMPTY, is the count of index entries possible for empty blocks that have been used (DSOP 3, 4, and 5).The count is the number of data blocks divided by 56, a ‘worst case’ estimate unlikely to occur because it presumes that every group of 56 blocks has at least one block that has been used but currently contains no records.

The next line, starting with *MOVED RECORDS, is the count of index entries possible for tracking moved records. However, because CA Datacom does not save a count of moved records, this value is currently always 0. Because it is normal for this to not be a large number, the estimate is expected to be reasonable for the RETIX estimate.

The last line is TOTAL and provides the sum of the key estimates from the prior lines. It also provides information about what is overriding the SORT= keyword. If the estimated count of keys from the record input is all zeros, the *DSOP values are not added into the total, and the override does not occur.

If the record counts are significantly incorrect but not zero, the sort uses the estimates as they exist. If the RETIX fails because of low sort count issues, change the JCL to provide a more accurate SORT= value then rerun the LOAD without using the SORTDFLT= keyword.

Example

For this example, database 2 was used. The first statement is a RETIX with KEYNAME=*SETR which recounts the records in the area. it was used only to ensure the counts were accurate in the example and is not intended for normal use.

```
RETIX  AREA=DD1,DBID=2,KEYNAME=*SETR
COMM  OPTION=CLOSE,DBID=2
INIT   DBID=2,AREA=IXX
RETIX DBID=2,SORT=99999,SORTDFLT=YES,
OPTION1='(I)'
```

The resulting output is as follows:

```
SORTDFLT INFORMATION
TABLE         RECORDS   KEYS         ESTIMATE
AGR               308      8            2,464
ALS             1,624      3            4,872
ARA                48      3              144
ATZ                 9      2               18
BAS                23      5              115
DVW                37      2               74
ELM               601      5            3,005
FCN                 0      3                0
FIL                86      2              172
FLD            11,845     10          118,450
HSD               202      1              202
JOB                 0      2                0
KEY               524     12            6,288
KWC                30      9              270
LIB                 0      2                0
MEM                 0      2                0
MOD                 0      2                0
OD                  0      3                0
PCV                 0      2                0
PER                 2      2                4
PGM                 0      2                0
PLN                 0      3                0
PNL                 0      2                0
PRC                 0      3                0
PRT                 0      2                0
REL             2,118      7           14,826
RPT                 0      2                0
STM                 0      3                0
STP                 0      2                0
SYS                12      2               24
TRG                 0      3                0
TXT                93      2              186
*DSOP CLUSTER                               0
*DSOP PARTIAL BLOCKS                       48
*DSOP EMPTY BLOCKS                          0
*MOVED RECORDS                              0
TOTAL                                 151,162   SORT=00000151162
```

Information from SORT:

```
CAS0000I  SORT    FILSZ=E151162,FIELDS=(1,186,BI,A),DYNALLOC=(SYSDA,03)
CAS0061I RCD IN 00000000,OUT 00149005
```

In this example, the count of keys sorted is less than the estimate because CA Datacom Datadictionary uses the rarely used NIL key feature to not be part of the index. NIL keys are defined to have a value of either all blanks (x'40') or all binary zeros (x'00').

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. A false wait condition can possibly occur when the Log Area is initialized with a lower TSN number than currently exists in the existing rows. To prevent this, assign a larger TSN number when initializing a Log Area. If TSN numbers are reused, the false wait condition occurs because a new transaction could possibly wait on another executing transaction that has the same number as an already committed transaction. This involves a short wait on the *wrong* transaction and is only a minor concern. It is not an integrity concern.

Resetting the TSN is desirable because the previously described error condition, involving the possibility of having duplicate TSNs, can produce a large overhead when running as part of a MUFplex. This only occurs and is a concern if the Log Area was initialized with a lower TSN than has been assigned. It is also possible if the TSN number 'wraps' going from 4 billion less one to 1, a very rare occurrence.

Resetting the TSN requests that the MUF zero every row TSN in the specified database and area that is committed. This includes multi-block reads of every block in the area. Every block with a row is written with multi-block writes or single block writes.

One *normal* condition could occur where a reset of the TSN should be used for all areas. When you are running a MUFplex, if the TSN number wraps from four billion back to one, the code then continues to execute properly but in wrap mode. This continues for the next 16 million transactions, which is typically a very long time. Before that situation occurs, it is best to reset the TSN for all the areas.

In addition to resetting the TSN, this form of RETIX can report the highest URI in the data area and reset the high URI value as stored and reported in the CSS if the current value is too low. MUF execution is optimized in full MUFPLEX with two MUFs enabled and running together.

Example

The following show the report that is currently generated:

RETIX AREA=A01,DBID=0997,KEYNAME=*RTSN

```
                 * DBID    997 AREA A01                REQUEST COMPLETE
ALL TSN'S RESET: YES
```

Values for ALL TSN's RESET are:

- **YES** - if the DBUTLTY function found no rows in the blocks as processed that had a row for an active transaction
- **NO** - if any row was subject to ROLBK as an active transaction
- **NOT URI** - if the area was not loaded as URI at the last LOAD

The report is not changed if the following conditions apply:

- The new solution is applied to DBUTLTY but not currently in the MUF
- The new solution is applied to the MUF but not currently in the DBUTLTY

If the new solution is applied to both the MUF and DBUTLTY, there are two cases when the report is enhanced.

Normal Case

The area had no row with a URI higher than the area high URI. The highest row URI is printed with the area high URI and a message that the high URI is unchanged.

Note: The high row URI may be less than the high area URI if the highest row URI values are for rows that have been deleted and those deletes have had the space reclaimed for new/moved rows.

```
                  * DBID    997 AREA A01                  REQUEST COMPLETE

 ALL TSN'S RESET: NO     HIGH ROW URI: X'000001F4'   HIGH URI: X'000001F4'   HIGH URI UNCHANGED
```

High URI

The second case is when the error exists that a row URI is higher than the area high URI and the area high URI is incremented to correct a defect. The following report shows that the new high URI as set.

Note: The new high URI as set is currently ‘rounded’ up (if needed) to a hexadecimal value ending with the last 10 bits set on.

```
                  * DBID    997 AREA A01                  REQUEST COMPLETE

ALL TSN'S RESET: YES    HIGH ROW URI: X'000001F4'   HIGH URI: X'000000FF'   HIGH URI SET X'000003FF'
```

MUF Connection when KEYNAME=*RTSN

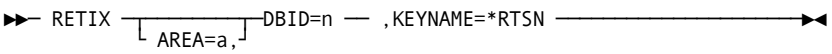
This function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

How to Use

When using the RETIX function to reset the TSNs, the MUF must be active. This utility function executes only in the MUF. 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)

Optional Keywords

AREA=

Identifies the data area for which the data space index is to be rebuilt or omitted to indicate that all areas in base are to be processed.

Valid Entries:

DATACOM-NAME of the area

Default Value:

All areas in the database

Chapter 39: RXXFIX (Recover End-Of-File)

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 MUF 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.

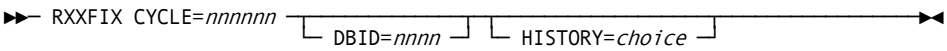
The RXXFIX function eliminates these problems. By specifying a cycle number from the RESTART process, the utility copies all the good data to a new tape before any possible I/O error, stops reading the suspect tape, and closes the new complete, non-redundant tape, if a spill was ongoing at the time of the failure.

Update SPILL History Table

The RXXFIX function can be used to update the SPILL history table by specifying the keywords DBID= and HISTORY=.

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)

Optional Keywords

DBID=*nnnn*

The DBID= keyword provides the database ID of the history database, where *nnnn* is the history database DBID.

Valid Entries:

the history database DBID, normally 1007

Default Value:

(No default)

HISTORY=*choice*

HISTORY=NO indicates that you want RXXFIX to do the CYCLE processing that is used either to fix an RXX with active recovery or for inactive recovery with the use of DISP=MOD (DISP= is the JCL disposition parameter). Therefore, specifying HISTORY=NO (or allowing HISTORY= to default to NO) indicates that you do not want to use RXXFIX to select a SPILL history table or to update the history information in a SPILL history table.

All allowed forms of YES (YES_RC0, YES-FAIL, and YES-RC4) are allowed and have the same meaning in RXXFIX.

Do not use RXXFIX to select or update SPILL history tables for RXX files that use spanned logging. If there is spanned logging in use, specify SPILL history tables using the SPILLOPT function as described in the section that follows. Do not ignore errors generated by using the SPILLOPT function, or correct the history record manually.

If the DBUTLTY RXXFIX function runs without finding any end of spill records, it ends successfully with an EOJ message but gives no indication (prints no lines in the report) showing that any end of spill action was taken.

Valid Entries:

NO, YES-FAIL, YES-RC0, or YES-RC4

Default Value:

NO

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//RXXIN    DD DSN=rxxinp,DISP=SHR                Recovery input file
//RXXOUT   DD DSN=rxxout,DISP=SHR                Recovery output file
//*
//SYSIN    DD *                                  Command Input
         RXXFIX    CYCLE=5
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                  CONTROL CARD(S)
                  .........1.........2.........3.........4.........5.........6.........7.........8
                  RXXFIX CYCLE=5

                  FUNCTION=RXXFIX
                     CYCLE=000005
```

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.

```
                  0000001   CYCLE COPIED
                  0000002   CYCLE COPIED
                  0000003   CYCLE COPIED
                  0000004   CYCLE COPIED
                  0000005   CYCLE COPIED
*** EOJ ***
```

This page of the report displays messages indicating the status of the Recovery File.

Chapter 40: SECURITY RESET (Clearing Security Validation Buffer)

The SECURITY RESET function is used to clear the security validation buffer. The SECURITY function requires that the MUF 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 reset 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 MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

When using a console command, these steps are unnecessary.

How to Use the Console Command

The MUF must be active when you 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

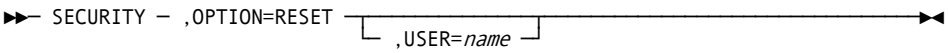
user

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, and Server 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

The MUF must be active when you 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, and Server 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command input
         SECURITY  OPTION=RESET
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
SECURITY OPTION=RESET

FUNCTION=SECURITY
   OPTION=RESET
```

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.

```
DB13105I - SECURITY RESET    REQUEST ACCEPTED
```

This page of the report displays a message indicating the success of the function execution.

Chapter 41: SET (Change Environment Processing)

The SET function allows you to change environment processing of options.

One common ability is to use ON-ERROR 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.

The different types of SET options are used to do the following:

- Control the style of error handling and remove an error condition.
- Assist in the debugging of a specific error condition that has occurred during a DBUTLTY execution.
- Override options for a specific DBUTLTY execution without changing the option for other DBUTLTY executions.
- Allow the testing of some interesting conditions. This use is intended to speed CA Technologies internal testing, but some users have asked for the ability as well, so that they can locally do QA on cases about which they have a concern.

No SET option communicates with MUF. All are local to this DBUTLTY execution.

Some SET options are NEVER allowed to occur when done as part of the DBIN1PR additional SYSIN processing. These are the following:

- DUMPPGM=

Some SET option can ONLY occur when done as part of the DBIN1PR additional SYSIN processing. Remember that DBIN1PR SET statements are required to have SET in columns 1-3 with one blank then the value. These are the following:

- CONSOLE_SECONDS=
- LINE=

Some SET options can occur at either input stream but must occur before functions other than SET, because they must occur before DBUTLTY configures itself for execution. These are the following:

- DATALN
- DATANO
- DXXNO
- IXXLN
- IXXNO

SET options not the listed just given work through either input stream. They can be changed during execution to change the method of execution for the functions that follow them, remaining in place until or unless a later SET option changes the setting or condition.

The SET function at either input stream supports stacking of options. This reduces console messages. They must be separated by a semicolon (;) as in this example: SET OPTION1=DATANO=999;DATALN=4096;DXXNO=999;IXXLN=4096. DBUTLTY supports no form of continuation of an individual keyword value.

The SET function having a value starting with an asterisk (*) is accepted and printed but then ignored.

The following console-like options are not specifically SET options. They are allowed to be used as a SET option through the SYSIN but not through DBIN1PR. They are individually documented with the console-like commands in the *CA Datacom/DB Database and System Administration Guide*.

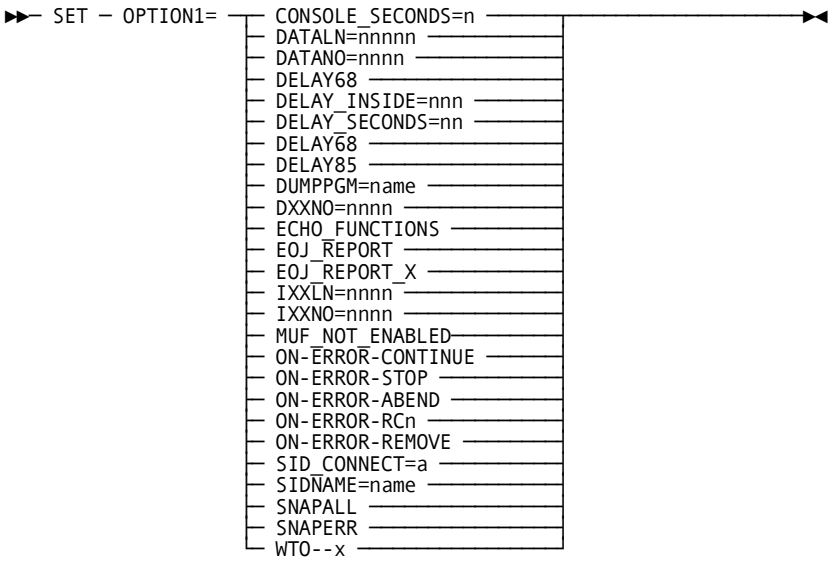
- DIAGOPTION n,y,x where ‘n’ is 0-23, ‘y’ is 0-255, and ‘x’ is ON/OFF/YES/NO
- DYNAMIC_EXTEND_MSGS OLD/TRK/CYL
- RCERROR x,y,nnnnn where ‘x’ is YES/NO/ON/OFF/FAIL, ‘y’ is YES/NO/ON/OFF/FAIL, and ‘nnnnn’ is a return code external/internal
- X_DEBUG x,n,y where ‘x’ is CBS/CL1/CDC/CXX/DSM/IO/OP1/PXX/LXX/RXX/UTM/COM/XX1/SQL1/XES/DST/XCF/CCI, ‘n’ is a hexadecimal 2-byte value of bits to turn on or off, and ‘y’ is YES/NO/ON/OFF
- X_OPEN_CLOSE_MSGS x where ‘x’ is YES/NO/ON/OFF
- X_TRACE_DSM n where ‘n’ is a number 0-32767

When to Use

All of these options are considered to be useful rarely, for special purposes. Review the list to know what options are available and use when appropriate.

How to Use

The following SET OPTION1= values are available in general. Some, however, have special rules about their use.



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:

Note: The DBUTLTY SET OPTION1= function in Version 12.0 handled DBSIDPR modules built by r10, r11, and Version 12.0 Macros, but the SET OPTION1= function in Version 14.0 handles only DBSIDPR modules built by the Version 14.0 Macro or the Version 12.0 Macro.

CONSOLE_SECONDS=*n*

Allows you to override the DBSYSID macro option CONSOLE_MINUTES to test the DBUTLTY automatic STATUS command. The *n* value can be 1-9. This is only an option through DBIN1PR.

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

DELAY_DBUTLTY_SECURITY

Allows you to override, in special cases, the DBSIDPR settings for the DBSYSID parameter DELAY_DBUTLTY_SECURITY=. It can be set to N or Y.

DELAY_INSIDE=*nnn*

This option speeds testing in QA systems. This option is used to test contentions between various functions and either each other or non-DBUTLTY opens. After the equal sign, specify a three-digit number from 000 through 999. The value is saved, and during selected functions a delay within the function's main-line processing a delay occurs to lengthen the execution of the function. Functions include the following:

- BACKUP
- EXTEND
- EXTRACT
- INIT
- LOAD
- REMOVE
- RECOVERY
- REORG
- RETIX
- SPILL

DELAY_SECONDS=*nn*

This option speeds testing in QA systems. After the equal sign, specify a two-digit number from 00 through 99. Processing the statement causing the DBUTLTY program to delay the number of seconds specified. It is used to change timing test cases, usually involving multiple concurrent DBUTLTY Job executions in parallel.

DELAY68

Allows you to override, in special cases, the DBSIDPR settings for the DBSYSID parameter DELAY68=. It can be set to 0 through 9 or 00 through 99.

DELAY85

Allows you to override, in special cases, the DBSIDPR settings for the DBSYSID parameter DELAY85=. It can be set to 0 through 9 or 00 through 99.

DUMPPGM=*name*

Allows you to specify the name of a program that DBUTLTY is to load into memory and format print it. This was built to assist support, ensuring content of modules relating to solutions. As an extension, if you want to allow either a DBSIDPR module name or an alternate name for a DBSYSID macro assembly, change to load and print in dump format when requested by CA Support.

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 CA 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.

The action once requested cannot be turned off.

EOJ_REPORT_X

This option is intended to only be used in consultation with CA 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.

The action once requested cannot be turned off.

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.

LINE=*aaa*

The value specified for *aaa* overrides the DBSIDPR parameter LINE=. You can specify for *aaa* either NEW or OLD.

LINE=OLD provides compatibility with previous (older) releases. For example, in older releases JCL could have been built for 121-byte output streams. LINE=OLD causes DBMUFPR and DBUTLTY to build and print lines at a length of 121.

LINE=NEW causes DBMUFPR and DBUTLTY to build and print lines with a length of 133.

For more information on LINE=, see the *CA Datacom/DB Database and System Administrator Guide*.

This option can only occur if provided through the DBIN1PR process.

MUF_ENABLED_OR_DISABLED

Allows a function to execute with MUF enabled or not enabled. We do not recommend this method for normal use. If the function is used when the MUF is not enabled, it blocks the MUF from enabling until the DBUTLTY execution completes. This method uses the SET function with OPTION1=MUF_ENABLED_OR_DISABLED. The use of this SET option is invalid if the DBUTLTY step has already connected to a MUF or opened the CXX locally.

This SET option removes any DELAY68 value set in determining if the MUF is enabled or not enabled. For functions that expect to execute with MUF, it is best that they are allowed to have the delay to wait for the MUF to enable.

MUF_NOT_ENABLED

Functions that normally execute with MUF enabled but can execute with the MUF not enabled exist for special situations where a MUF cannot enable due to restart errors. Such restart errors have to be (or can be) corrected before MUF successfully enables. These functions can run in the same DBUTLTY step as a function that has to run with MUF not enabled, or they have to declare the intent to run this complete DBUTLTY step with MUF not enabled. To do this, the SET function with OPTION1=MUF_NOT_ENABLED is used. The use of this SET option is invalid if the DBUTLTY step has already connected to a MUF or opened the CXX locally. The function converts all following functions from being able to run with MUF not enabled to being required to run with MUF not enabled for this execution of DBUTLTY. This ability is not expected or recommended for use by functions that cannot complete quickly. These functions are explained in another section.

This SET option removes any DELAY68 value set in determining if the MUF is enabled or not enabled. For functions that expect to execute with MUF, it is best that they are allowed to have the delay to wait for the MUF to enable.

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. For example, in z/OS one such error is a B37 abend.

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.

SID_CONNECT=

This option exists to speed testing in QA systems. After the equal sign you can specify a letter L (for LOCAL), X (for XCF), or C (for CCI). This is similar to what exists in the DBSYSID Macro used to generate a DBSIDPR module, keyword CONNECT_ALLOW_PRIORITY=. This testing option is not edited (checked) for valid letters. It is always accepted but will only have affect if done before the first function that attempts to connect to MUF.

SIDNAME=*name*

When a DBSIDPR is used by DBUTLTY that has a TARGET_MUF_LIST= with multiple entries, most functions work exactly like a user program would, and DBUTLTY finds the MUF locally or remotely and executes against it. Some functions target the MUF and do not have to run on the *right* LPAR for success. For example, if two MUFs are enabled and a DBUTLTY EOJ is executed, it ends the MUF that is local to the LPAR on which DBUTLTY executes unless it contains neither MUF, in which case it EOJs either of the MUFs, based upon the first MUF found. One solution is to route all DBUTLTY functions that need a specific MUF to the proper LPAR. Another solution is to force the DBUTLTY function to the desired MUF. This is done with a SET function with the keyword OPTION1=SIDNAME=*name,* naming the member in a library with DBSYSID macro assembly results that has specific MUF names instead of the standard DBSIDPR. Some functions need to execute at each executing MUF in a MUFplex such as ACCESS. For those cases, it is necessary to use a specific single MUF and execute ACCESS against it if enabled and then looping through all the candidate MUFplex MUFs that are enabled.

SNAPALL

This diagnostic tool is available, when requested by CA 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. This option cannot be turned off.

SNAPERR

This diagnostic tool is available, when requested by CA 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 CA Support with the additional information needed to find a solution. This option cannot be turned off.

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.

Valid Entries:

as shown in the list above

Default Value:

OPTION1=ON-ERROR-STOP
OPTION1=ON-ERROR-ABEND

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command Input
           SET OPTION1=ON-ERROR-STOP
           SET OPTION1=ON-ERROR-CONTINUE
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
   SET OPTION1=ON-ERROR-STOP

FUNCTION=SET
   OPTION1=ON-ERROR-STOP
```

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.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
   SET OPTION1=ON-ERROR-CONTINUE

FUNCTION=SET
   OPTION1=ON-ERROR-CONTINUE
```

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.

Chapter 42: SPILL (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)).

The SPILL function transfers data from the Log Area (LXX) to the Recovery File (RXX). All spilling is controlled and done through the MUF, which must be enabled to do a spill. The SPILL function is allowed for both active and inactive recovery.

If blocks are spillable, the utility opens the Recovery File, copies the appropriate log blocks (from MUF) 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 MUF startup options.

Note: When using inactive recovery (that is, NO specified for the MUF 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 can be run at any time to request that the MUF 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 date/time 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 MUF 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.

Dual Recovery File (RXX) Support

Dual Recovery File (RXX) support is available for sites that use inactive recovery. This provides the ability to create two copies of the RXX with one pass of the Log Area (LXX). To allow for a dual Recovery File, a dual RXX parameter has been added to the LOGRCV MUF startup option.

Note: For more information on the LOGRCV MUF startup option, see the *CA Datacom/DB Database and System Administration Guide*.

MUF Connection

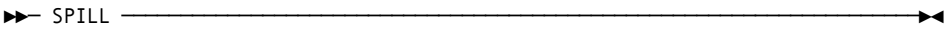
This function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

How to Use

The MUF must be active when you 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. In the line that shows //RXX DD ... the three dots (...) represent the desired DD statement options. Do not use the DCB= option, however, because CA Datacom/DB forces the appropriate values to be used, and those appropriate values override any DCB= values set by users.

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
//RXX      DD ...                                Recovery data set (output)
set
//SYSIN    DD *                                  Command Input
         SPILL
/*
```

The following shows sample JCL when the MUF option is set for dual RXX output. In the line that shows //RXX1 DD ... and //RXX2 DD ... the three dots (...) represent the desired DD statement options. Do not use the DCB= option, however, because CA Datacom/DB forces the appropriate values to be used, and those appropriate values override any DCB= values set by users. Using dual RXX output, you could if desired direct RXX1 to disk and RXX2 to 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.

```
//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
//RXX1     DD ...                                Recovery data set 1 (output)
//RXX2     DD ...                                Recovery data set 2 (output)
//SYSIN    DD *                                  Command Input
         SPILL
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
                    CONTROL CARD(S)
                    .........1.........2.........3.........4.........5.........6.........7.........8
                    SPILL

                    FUNCTION=SPILL
```

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.

```
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

DB00404I - RXX LAST RECORD DATE/TIME -  ccyy/mm/dd/ hh.mm.ss
```

This page of the report displays messages about the success of the spill. See the *CA Datacom/DB Message Reference Guide* for details on each message.

Chapter 43: SPILLOPT (Transfer Data to RXX Using MAX/MIN)

The SPILLOPT SPILL= function allows you to specify MAX or MIN when requesting a spill.

Dual Recovery File (RXX) support is available for sites that use inactive recovery. See Dual Recovery File (RXX) Support (see page 864).

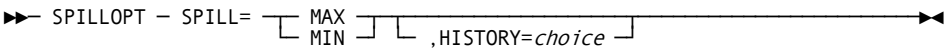
Update SPILL History Table

The DBUTLTY SPILLOPT function can be used to update the SPILL history table by specifying the keyword HISTORY=.

Note: For details about the fields of the SPILL history table, see the SPILL history table section in creating and maintaining the history database chapter in the *CA Datacom/DB Database and System Administration Guide*.

How to Use

The MUF must be active when you 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

,HISTORY=*choice*

Specify NO for the HISTORY= keyword (or allow it to default to NO) to indicate that you do not want to select a SPILL history table with SPILLOPT or that you do not want the history information in a SPILL history table to be updated.

In addition to requesting the history, specifying YES-FAIL, YES-RC0, or YES-RC4 for HISTORY= indicates what action you want taken if a problem is encountered in handling the history.

If you specify HISTORY=YES-FAIL, it indicates that you are requesting the history record and, if any of the commands fail, that you want the spill and the DBUTLTY function in error to be terminated immediately.

If you specify HISTORY=YES-RC0, it indicates that you are requesting the history record but, if it cannot be done, to issue a message and complete the spill without ending the DBUTLTY function in error. You would then have to correct the problem for future executions and execute RXXFIX to build or update the appropriate history spill record.

Valid Entries:

NO, YES-FAIL, YES-RC0, YES-RC4

Default Value:

NO

,OPTION2=FORCERXX

Indicates that you want an empty RXX dataset written even if there are no records in the LXX that are subject to a SPILL. The RXX dataset is built as an empty dataset and if SPILL history is enabled, there is no SPILL history record created. The DBUTLTY RECOVERY functions can read the new empty RXX dataset without error. The following console message is received:

DB 13083I - LOG AREA EMPTY, SPILL IS NOT NECESSARY.

Valid Entries:

,OPTION2=FORCERXX

Default Value:

Not specified. No empty RXX dataset is created.

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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//RXX      DD DSN=rxx.data.set,DISP=SHR,          Recovery data set
//            UNIT=tape                             (output)
//SYSIN    DD *                                   Command Input
         SPILLOPT SPILL=MIN
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
SPILLOPT SPILL=MIN

FUNCTION=SPILLOPT
   SPILL=MIN
```

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.

```
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

DB00404I - RXX LAST RECORD DATE/TIME -  ccyy/mm/dd/ hh.mm.ss
```

This page of the report displays messages about the success of the spill. See the *CA Datacom/DB Message Reference Guide* for details on each message.

Chapter 44: UNLOCK (Unlock a Data Area)

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) for information about this function.

When to Use

Use the UNLOCK function to reverse a LOCK function.

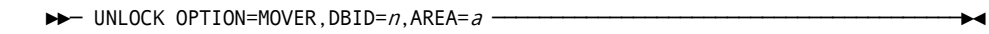
This function communicates with one MUF. Be aware of the following:

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

If the z/OS Cross-System Coupling Facility (XCF) is being used, ensure that the TOGROUP DBSIDPR parameter is correctly defined. See the *CA Datacom/DB Database and System Administration Guide* for more information on XCF.

How to Use

The MUF must be active when you 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.

```
//jobname    See the previous note and JCL Requirements.
//       EXEC PGM=DBUTLTY,REGION=2M
//STEPLIB    See the previous note and JCL Requirements.
//SYSIN    DD *                                  Command Input
     UNLOCK OPTION=MOVER,DBID=1,AREA=DEM
/*
```

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
CONTROL CARD(S)
.........1.........2.........3.........4.........5.........6.........7.........8
UNLOCK OPTION=MOVER,DBID=1,AREA=DEM

FUNCTION=UNLOCK
   OPTION=MOVER
   DBID=1
   AREA=DEM
```

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.
- If there are any messages related to syntax processing, they are displayed following the analysis of keywords.

```
*                                    REQUEST COMPLETE
```

This page of the report displays a message indicating that the UNLOCK completed successfully.

Chapter 45: 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.

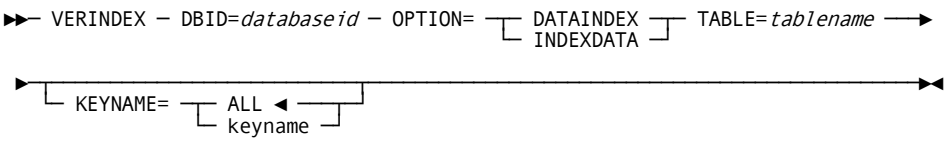
When to Use

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 Datadictionary 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.

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

Sample Report: DATAINDEX without an Error

```
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
```

This page of the report shows the DATAINDEX function without an error.

Sample Report: DATAINDEX with an Error

```
 KEY TST02 NO INDEX FOUND, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 00004478 DATA TTTR 00000106
 00000000 TSN 00001111  C2875E07 BE0F7C83                                          *Bg
...@c
```

This page of the report shows the DATAINDEX function with an error.

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 Datadictionary 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.

Sample Report

Following is a sample report page. For an example report header, see Sample Report Headers.

```
FUNCTION=VERINDEX
                      DBID=00997
                      OPTION=INDEXDATA
                      TABLE=K01

KEY RELATED STATISTICS:
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)
```

This page of the report shows the following under KEY RELATED STATISTICS:

KEYID

Displays the identification number of key.

KEYNAME

Displays the name of key.

MAST

Identifies whether the key is a Master Key.

MATCH_COUNT

Displays the count of index entry key values found that matched data record key values.

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 provided in the SYSOUT data set INDOUBT. 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 ROW INDEX MISMATCH, INDEX VALUE BELOW IN SYSUDUMP FORMAT, URI 0001E997 DATA TTTR 00004103 TSN 00001111
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   *24336614805362880542939600007575*
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.

If not found, the INDOUBT SYSOUT is dynamically allocated. It appears as //INDOUBT DD SYSOUT=*. If many errors are expected, as in a very large controlled test of mismatching data and index, the size can be limited with the OUTLIM option set: OUTLIM=999999.