Progress®



OpenEdge® Data Management:
Database Administration

OpenEdge®

Copyright

© **2019 Progress Software Corporation and/or its subsidiaries or affiliates. All rights reserved.**

These materials and all Progress® software products are copyrighted and all rights are reserved by Progress Software Corporation. The information in these materials is subject to change without notice, and Progress Software Corporation assumes no responsibility for any errors that may appear therein. The references in these materials to specific platforms supported are subject to change.

Corticon, DataDirect (and design), DataDirect Cloud, DataDirect Connect, DataDirect Connect64, DataDirect XML Converters, DataDirect XQuery, DataRPM, Deliver More Than Expected, Icenium, Kendo UI, Kinvey, NativeScript, OpenEdge, Powered by Progress, Progress, Progress Software Developers Network, Rollbase, SequeLink, Sitefinity (and Design), Sitefinity, SpeedScript, Stylus Studio, TeamPulse, Telerik, Telerik (and Design), Test Studio, and WebSpeed are registered trademarks of Progress Software Corporation or one of its affiliates or subsidiaries in the U.S. and/or other countries. Analytics360, AppServer, BusinessEdge, DataDirect Autonomous REST Connector, DataDirect Spy, SupportLink, DevCraft, Fiddler, JustAssembly, JustDecompile, JustMock, Kinvey Chat, NativeChat, NativeScript Sidekick, OpenAccess, ProDataSet, Progress Results, Progress Software, ProVision, PSE Pro, SmartBrowser, SmartComponent, SmartDataBrowser, SmartDataObjects, SmartDataView, SmartDialog, SmartFolder, SmartFrame, SmartObjects, SmartPanel, SmartQuery, SmartViewer, SmartWindow, and WebClient are trademarks or service marks of Progress Software Corporation and/or its subsidiaries or affiliates in the U.S. and other countries. Java is a registered trademark of Oracle and/or its affiliates. Any other marks contained herein may be trademarks of their respective owners.

March 2019

Last updated with new content: Release 12.1.0

Updated: 2019/04/29

Table of Contents

Preface....29
Purpose....29
Audience....29
Organization....30
Using ABL documentation....32
References to ABL compiler and run-time features....32
References to ABL data types....32
Typographical conventions....33
Examples of syntax descriptions....34
Long syntax descriptions split across lines....35
Complex syntax descriptions with both required and optional elements....36
OpenEdge messages....36
Obtaining more information about OpenEdge messages....37

Part I: Database Basics....39

Chapter 1: Creating and Deleting Databases....41
Ways to create an OpenEdge database....41
Database creation and large file support....42
Creating a database with PROSTRCT CREATE....43
Creating a structure description file....43
Create a database structure extent....49
Creating a database with the PRODB utility....51
PRODB maintains pathname convention....52
Creating a database with a Data tool....52
Converting an OpenEdge Release 11 Database to OpenEdge Release 12....53
Using the Schema Mover....54
Converting an OpenEdge Release 10 database to OpenEdge Release 12....56
Database conversion utilities....56
Copying a database....56
Copying a database using PROCOPY....57
Deleting a database....58

Chapter 2: OpenEdge RDBMS Limits....59
Database block sizes....60
Maximum number of records per block....60
Number and size of storage areas....60

Recovery sizes 65
Table and index limits....65
Number and size of sequences....66
Maximum size of the primary recovery (BI) area....66
Maximum database size....67
Number of connections per database....67
Number of simultaneous transactions per database....67
OpenEdge database name limits....68
File Handles....68
Shared memory....69
Shared memory interactions....69
Data types and values....70

Chapter 3: Starting Up and Shutting Down....73

Unified Broker Framework....73
OpenEdge Management....74
OpenEdge Explorer....74
Managing database configurations 74
Using the DBMAN command-line utility 75
Starting a server or broker....76
Specifying international character sets....76
Network addressing using -S and -H....77
Starting multiple brokers using the same protocol....77
Accessing a server behind a firewall....78
Starting database brokers for SSL connections....79
Threaded Database Server Functionality Delivered on Windows NT64, AIX64, and Linux64 Platforms....80
Initiating a Threaded Server Process....80
Monitoring Threaded Servers and Remote Connections....81
Starting and stopping background writers....81
Starting and stopping an APW....81
Starting and stopping a BIW....82
Starting and stopping an AIW....82
Stopping a server or broker....82
PROSHUT command....83
PROMON shut down database option....85

Part II: Protecting Your Data....87

Chapter 4: Backup Strategies....89

Identifying files for backup....89
Determining the type of backup....90
Full backups....91

Incremental backups....91
Online backups....93
Offline backups....93
Choosing backup media....93
Creating a backup schedule....94
Database integrity....94
Database size....94
Time....95
Unscheduled backups....95

Chapter 5: Backing Up a Database....97

Using PROBKUP....98
Performing an online full backup with PROBKUP....98
Testing backups....100
Archiving backups....100
Performing an offline backup....101
Performing an online backup....102
Improving performance of an online backup....102
Using database quiet points....102
Performing an operating system backup....104
Database backup examples....105
Incremental backup example....105
Full backup example....107
Verifying a backup....110
CRC codes and redundancy in backup recovery....111
CRC codes....111
Error-correction blocks....111
Restoring a database....111
Using the PROREST utility to restore a database....112
Important rules for restoring backups....113
Obtaining storage area descriptions using PROREST....113
Database restore examples....114

Chapter 6: Recovering a Database....117

Introduction to recovery mechanisms....118
Crash recovery....118
Roll-forward recovery....120
Two-phase commit....121
File locations that ensure safe recovery....122
Developing a recovery plan....122
Time needed for recovery....122
Recovery guidelines....123
Sample recovery plans....123

Example 1: Low availability requirements....123
Example 2: Moderate availability requirements....124
Example 3: Moderate-to-high availability requirements....124
Example 4: high availability requirements....125
Sample recovery scenarios....126
After-imaging and roll-forward recovery commands....129
Recovering from system failures....130
System crash while running RFUTIL ROLL FORWARD....130
System crash while running other utilities....130
System crash while backing up the database....131
System crash while database is up....131
Recovering from media failures....131
Loss of the DB files, BI files, or both....131
Loss of the AI file....132
Loss of database backup....132
Loss of transaction log file....133
Recovering from a full disk....133
After-image area disk....134
Control or primary recovery area disk....134
Transaction log file disk....134
Truncating the BI file....135
Releasing shared memory....135
Recovering from a lost or damaged control area 136
Unlocking damaged databases....137
Dumping tables from a damaged database....137
Forcing access to a damaged database 138

Chapter 7: After-imaging....139

After-image areas and extents....140
Fixed-length extents....140
Variable-length extents 140
Estimating after-imaging disk space requirements....141
Creating after-image areas....141
Adding AI areas to an existing database....142
Enabling after-imaging offline....142
Enabling encrypted after-imaging offline for an encryption-enabled database....143
Enabling unencrypted after-imaging offline for an encryption-enabled database....143
Enabling after-imaging online....143
Enabling encrypted after-imaging online for an encryption-enabled database....144
Enabling unencrypted after-imaging online for an encryption-enabled database....144
Managing after-imaging files....144
Monitoring AI file status....144
Switching to a new AI file....146
Archiving an AI file 147

Making an AI file available for reuse....149
AI File Management utility....150
Automatic extent archiving....150
Enabling your database for automated AI file management....152
Monitoring and adjusting automated AI File Management....153
Restarting the AI file management utility daemon....154
Archived extents....154
Archive log file....155
Database recovery with an archival log....159
Add and reorder AI extents....159
Performing roll-forward recovery....161
Roll forward with the OPLOCK qualifier....161
Rolling forward multiple files....162
After-image sequences....164
Sequence not required....165
Sequence required....165
Disabling after-imaging....166
Disabling encryption of AI files when disabling after-imaging....167

Chapter 8: Maintaining Security....169

Establishing an OpenEdge user ID and password....170
OpenEdge user IDs....170
OpenEdge passwords....171
Validating an OpenEdge user ID and password....172
Establishing authentication for your OpenEdge database....172
Tables used from ABL only....172
Tables used from SQL only....173
Tables used from both ABL and SQL....173
Connection security....173
Designating valid users in the _User table accounts....174
Designating a security administrator....175
Deleting a user from the _User table accounts....176
Changing a password for database users....176
Running a user report....176
Schema security....177
Freezing and unfreezing schema....177
Operating systems and database security....177

Chapter 9: Auditing....179

Auditable events....179
Audit events....180
Security events....180
Schema events....181

Data events....181
Administration events....181
Utility events....181
User events....182
Application events....182
Auditing states....182
Enabling and disabling auditing....183
Enabling auditing....183
Re-enabling auditing from a deactivated state....184
Disabling auditing....184
Auditing tables....185
Indexes on auditing tables....186
Archiving audit data....187
Audit archive process....187
Reconfiguring audit areas....189
Split one audit area into separate data and index areas....190
Combine audit data and index areas into one area....190
Move existing audit data and indexes to new areas....191
Purge data in existing audit data areas....192
Auditing impact on database resources....193
Auditing impact on database utilities....193
Identifying the privileged user....193
Utility modifications....196
Protected tables....196
Protected areas....196
Enhanced utilities....196
Expanded PROUTIL syntax....197

Chapter 10: Transparent Data Encryption....199

Overview....199
Before you start....200
What is encryptable....201
Data considerations....201
Database key store....201
OpenEdge supported ciphers....203
Enabling Encryption....204
Adding an encryption policy area....204
Running ENABLEENCRYPTION....205
Enabling BI file encryption after enabling encryption....206
Enabling AI file encryption after enabling encryption....206
Verifying enablements....207
Creating encryption policies....208
Creating encryption policies with PROUTIL EPOLICY....208
Creating encryption policies with Data Admin....209

Encrypting your existing data....209
EPOLICY MANAGE UPDATE EXAMPLE....209
All new data is encrypted....210
Running with Transparent Data Encryption enabled....211
Manual start....211
Autostart....212
Restricted utility access....213
Database backups....214
Moving tables and indexes....214
Binary dump and load of data....217
Auditing Transparent Data Encryption....218
Replication support....222
Maintaining Transparent Data Encryption....222
Rekey encryption policies with Data Admin....222
Rekey encryption policies with PROUTIL EPOLICY....224
Viewing encryption policy history in Data Admin....224
Disabling encryption....225
OpenEdge SQL support for Transparent Data Encryption....227

Chapter 11: Replicating Data....229

Replication schemes....229
Trigger-based replication....229
Log-based site replication....230
Asynchronous replication....230
Database ownership models....230
Distribution model....230
Consolidation model....231
Implementing log-based site replication....232
Log-based replication procedure....232

Chapter 12: Failover Clusters....235

Overview....236
Related software and hardware....236
Installation....237
Configuration....237
Security....239
Performance....239
Logging....239
Required cluster components....240
Network considerations....241
Terms and concepts....242
Resources and dependencies....242
Failure and recovery action....242

Fail over policies....242
Using the PROCLUSTER command-line interface....243
Cluster-enabling a database....243
Disabling a cluster-enabled database....244
Starting a cluster-enabled database....245
Stopping a cluster-enabled database....245
Terminating a cluster-enabled database....245
Isalive and looksalive....246
Results of enabling an OpenEdge database for fail over....246
Database UUID file (HPUX 64 bit only)....247
Changing the structure of the database....247
Adding extents on a volume group or file system different from the database (AIX only)....247
Platform-specific considerations....247
Adding nodes where the database can be run for AIX....248
Upper limit on the number of packages for HPUX....248
Directory of registered packages for HPUX....248
Restarting on the same node for HPUX....248
Sun Cluster uses global devices....249
Windows Server 2008 R2 Cluster support with OpenEdge Windows 64-bit....249
Using a cluster-enabled database with the AdminServer....249
Using a cluster-enabled database with standard commands....250
Using the Windows Cluster Administrator....250
Starting a database resource with the Cluster Administrator....251
Shutting down a database resource with the Cluster Administrator....251
Adding a secondary broker....251
Emergency disabling of a cluster-enabled database....252
UNIX cluster management commands....252

Chapter 13: Distributed Transaction Processing....255

Distributed transactions....256
Two-phase commit with ABL clients....256
How the database engine implements two-phase commit....256
Two-phase commit and roll-forward recovery....258
Enabling two-phase commit....259
Modifying the database nickname and priority....259
Transaction log area....260
Deactivating two-phase commit....260
Limbo transactions with two-phase commit....261
Resolving limbo transactions....261
Resolving limbo transactions with PROUTIL....264
Resolving limbo transaction scenarios....266
Two-phase commit case study....267
Java Transaction API (JTA) support....269
JTA resource impact....269

JTA processing impact....269
Enabling JTA support....270
Disabling JTA support....270
JTA transactions and OpenEdge Replication....270
Monitoring JTA transactions....270
Resolving JTA transactions....271

Part III: Maintaining and Monitoring Your Database....273

Chapter 14: Managing Performance....275

Introduction to performance management....276
Tools for monitoring performance....276
PROMON utility....276
Virtual system tables....276
Server performance factors....277
CPU usage....277
Disk I/O....277
Database I/O....278
Before-image I/O....288
After-image I/O....293
Direct I/O....295
Memory usage....296
Process address space overview....296
Controlling memory use....297
Shared memory allocation....297
Monitoring memory use....298
Shared memory summary....298
Alternate Buffer Pool....298
Overview....299
Designating objects to the Alternate Buffer Pool....299
Viewing Alternate Buffer Pool selections....301
Allocating Alternate Buffer Pool....303
Operating system resources....303
Processes....303
Semaphores....303
Spin locks....306
File descriptors....306
Network Communication....306
Prefetch tuning....306
Message wait....307
Database fragmentation....308
Analyzing database fragmentation....308
Eliminating database fragmentation....309

Managing fragmentation....309
Index use....311
Analyzing index use....311
Compacting indexes....312
Index utilities....313
Rebuilding indexes....315
Activating a single index....320
Database Client Notification....321
Changing Database Client Notification poll time....322
Virtual system tables....323

Chapter 15: Maintaining Database Structure....325

OpenEdge Structure utility....326
OpenEdge Structure Statistics utility....326
OpenEdge Structure List utility....328
Using the OpenEdge Structure Add utility....329
PROSTRCT ADD and pathnames....330
Adding extents to existing storage areas....331
OpenEdge Structure Add Online utility....332
Area numbers....335
Area number assignment....335
Trimming unused area memory....338
Error checking....340
Validating structure description files....341
Using the OpenEdge Structure Remove utility....342
Maintaining indexes and tables....343
Moving tables....343
Moving indexes....344
Compacting indexes....344
Performing chain analysis online....345
Important considerations when performing chain analysis online....346
Using virtual system tables....346

Chapter 16: Multi-tenant database....347

Multi-tenant database introduction....347
Multi-tenant database architecture....348
Simple tenancy....349
Multi-tenant groups....349
Multi-tenant features....350
Enabling multi-tenancy....351
Creating multi-tenant elements....351
Database administration tools....351
ABL....353

OpenEdge SQL....353
Tenant Data Recovery....353
Restrictions....354
Step 1: Restoring your database backup....355
Step 1A: restricting access to your temporary database....356
Step 2: Rolling forward after-image files....356
Step 3: Moving data with PROUTIL DATAMOVE....358
Move multiple tables and sequences with the DATAMOVE user interface....359
Step 4: Tenant data recovery clean up....366
Database utilities....367
Unchanged utilities....368
Indirectly changed utilities....368
Multi-tenant aware utilities....368
Multi-tenant index rebuild....369
Controlling the usage of multi-tenant database resources using multi-tenant governors.370

Chapter 17: Table-partitioned database....371

Table partitioning details....372
Table partition types....372
Table partitioning restrictions....374
Indexes for table partitions....375
Partitioning your database....375
Managing partitions....376
Splitting partitions....376
Merging partitions....377
Truncating partitions....378
Renaming partitions....378
Altering partition state....379
Viewing partitions....379
Copying partition data....380
Partition-level database administration....387
Partition-scoped utilities....387
Table-partitioned index rebuild....388

Chapter 18: Change Data Capture....389

Change Data Capture highlights....389
CDC and other RDBMS features....391
Enabling CDC....396
Using the CDC cache....396

Chapter 19: Diagnostic Data Collection....399

Events....400
Specifying an event....400

Event Levels....400
Data Files....402
Protrace data....405
DiagProc file....405
Diagnostic data startup parameters....405
Diagnostic data collection in PROMON....406
Diagnostic Data Collection menu....406
Diagnostic data output file formats....407
Diagnostic tracking file....407
Diagnostic BI threshold files....407
Lock table overflow files....409
Syserror event files....410

Chapter 20: Dumping and Loading....411

Overview of dumping and loading....412
Dump and load limitations....412
Dumping ABL database definitions....412
Dumping ABL database or table definitions....413
Definition file format....414
Multi-tenant keywords....415
Creating an incremental ABL data definitions file....415
Dumping sequence definitions....416
Dumping auto-connect records....416
Dumping Change Data Capture policies....417
Loading Change Data Capture policies....418
Dumping database contents....418
Dumping table contents with PROUTIL....419
Dumping selective contents with PROUTIL....420
Dumping table contents with a Data tool....422
Contents file format....423
Dumping sequence values with a Data tool....424
Dumping user table contents with a Data tool....424
Dumping an SQL view file's contents with a Data tool....425
Loading database definitions....425
Loading updated ABL data definitions....426
Loading database contents....426
Loading table contents in binary format with PROUTIL....427
Loading table contents with a Data tool....428
Loading user table contents with a Data tool....429
Loading an SQL view file contents....429
Loading sequence values using a Data tool....430
Bulk loading....430
Creating a Bulk Loader description file....430
Modifying a Bulk Loader description file....431

Loading table contents with the Bulk Loader qualifier 432
Reconstructing bad load records....433
Specialized dump and load techniques....433
Creating a starting version of a database....433
Using a constant table in multiple databases....434
Economizing disk space....434
Optimizing data for sequential access....435
Optimizing data for random access....436
Using no-integrity mode bulk data loading....436
Saving disk space by deactivating indexes....437

Chapter 21: Logged Data....439
OpenEdge database log file....439
Database Log file customization....441
Managing log file size....441
Log file archive and truncate....441
Log file truncate and archive startup parameters....442
PROMON Database Log file management....443
_DbParams VST....443
Archiving....443
Truncated and archived files....444
Saving key database events....445
Defining key events....445
Saving key events....446
Enabling save key events....446
Disabling save key events....447
Stored key events....447
_KeyEvt table....452
Event logging in Windows 453
Managing OpenEdge RDBMS events in Windows....454
Understanding the event log components....454
The Event Log and the registry....455
Client database-request statement caching....456

Part IV: Reference....459

Chapter 22: Startup and Shutdown Commands....461
Startup command syntax....461
Database startup and shutdown commands....462
PROAIW command 463
PROAPW command....463
PROBIW command....464
PROQUIET command....465

PROSERVE command....467
PROSHUT command....468
PROWDOG command....471

Chapter 23: Database Startup Parameters....473

Issuing startup parameters....474
Database server performance parameters....474
Database server-type parameters....478
Database server internationalization parameters....478
Database server statistics parameters....479
Database server consistency check parameters....479
Database server SSL connection parameters....480
Database server network parameters....480
Startup parameter usage categories....482
Alphabetical listing of database startup parameters....482
AdminServer Port (-adminport)....482
After-image File Management Archive Directory List (-aiarcdir)....483
After-image File Management Archive Directory Create (-aiarcdircreate)....483
After-image File Management Archive Interval (-aiarcinterval)....484
After-image Buffers (-aibufs)....484
After-image Stall (-aistall)....485
Area consistency check (-AreaCheck)....485
Blocks in Database Buffers (-B)....486
Blocks in Alternate Buffer Pool (-B2)....486
Base Index (-baseindex)....487
Base Table (-basetable)....487
Base user index (-baseuserindex)....488
Base user table (-baseusertable)....488
Before-image Buffers (-bibufs)....489
Threshold Stall (-bistall)....489
Recovery Log Threshold (-bithold)....490
Schema Cache File (-cache)....490
CDC cache size (-cdcsize)....491
SQL Server Java Classpath (-classpath)....492
Cluster Mode (-cluster)....492
Conversion Map (-convmap)....493
Case Table (-cpcase)....493
Collation Table (-cpcoll)....493
Internal Code Page (-cpinternal)....494
Log File Code Page (-cplog)....494
Print Code Page (-cpprint)....495
R-code in Code Page (-cprcodein)....495
Stream Code Page (-cpstream)....496
Terminal Code Page (-cpterm)....497

Crash Recovery Status (-crStatus)....497
Crash Recovery Transaction Display (-crTXDisplay)....498
Database consistency check (-DbCheck)....498
Direct I/O (-directio)....499
Encryption cache size (-ecsize)....499
Enforce Mm (-enforceMm)....500
Event Level (-evtlevel)....500
Before-image Truncate Interval (-G)....501
Group Delay (-groupdelay)....501
Host Name (-H)....502
Hash Table Entries (-hash)....502
No Crash Protection (-i)....502
Index consistency check (-IndexCheck)....503
Index Range Size (-indexrangesize)....504
Internet Protocol (-ipver)....505
Key Alias (-keyalias)....506
Key Alias Password (-keyaliaspasswd)....506
Lock Governor (-LGovernor)....506
Lock Table Entries (-L)....507
Lock Table Hash Size (-lkhash)....508
Lock release (-lkrela)....508
LRU force skips (-lruskips)....508
LRU alternate buffer pool force skips (-lru2skips)....509
Auto Server (-m1)....509
Secondary Login Broker (-m3)....510
Maximum Clients per Server (-Ma)....510
Maximum area number (-maxAreas)....510
Maximum Dynamic Server (-maxport)....511
Maximum JTA Transactions (-maxxids)....511
Memory consistency check (-MemCheck)....512
Delayed BI File Write (-Mf)....512
Minimum Clients per Server (-Mi)....513
Minimum Dynamic Server (-minport)....513
Message Buffer Size (-Mm)....514
Maximum Servers (-Mn)....514
Servers per Protocol (-Mp)....515
Maximum Servers per Broker (-Mpb)....515
User MUX Latches (-mux)....516
Shared-memory Overflow Size (-Mxs)....516
Network Type (-N)....517
Number of Users (-n)....517
Nap Time Steps (-napstep)....518
Login Governor (-nGovernor)....518
Message Wait (-Nmsgwait)....519
No Session Cache (-nosessioncache)....519

Number of checkpoint statistics (-numcheckpointstats)....519
Storage Object Cache Size (-omsize)....520
Prompt for Passphrase (-Passphrase)....520
Pending Connection Time (-PendConnTime)....520
Parameter File (-pf)....521
Database Service Communication Area Size (-pica)....521
Pin Shared Memory (-pinshm)....522
Prefetch delay (-prefetchDelay)....522
Prefetch factor (-prefetchFactor)....522
Prefetch Num Recs (-prefetchNumRecs)....523
Prefetch Priority (-prefetchPriority)....523
Configuration Properties File (-properties)....524
Buffered I/O (-r)....524
Record free chain search depth factor (-recspacesearchdepth)....524
Service Name (-S)....525
Schema Lock Wait Queue(-schlockwq)....526
Security cache size (-secsize)....526
Semaphore Sets (-semsets)....527
Server Group (-servergroup)....528
Session Timeout (-sessiontimeout)....528
Shared memory segment size (-shmsegsize)....529
Spin Lock Retries (-spin)....530
Lock Wait Timeout (-SQLLockwaitTimeout)....531
SQL rowid identifier(-SQLQuotedRowid)....531
Authorized Data Truncation(-SQLTruncateTooLarge)....532
Autonomous Schema Update (-SQLWidthUpdate)....532
Autonomous Update Statistics (-SQLAutoStats)....533
Server-side JOIN (-ssj)....533
SSL (-ssl)....534
STS logging level (-stslogginglevel)....534
Temporary Directory (-T)....535
Table consistency check (-TableCheck)....536
Table Range Size (-tablerangesize)....537
Threaded Server (-threadedServer)....537
Threaded Server Stack (-threadedServerStack)....538
Transaction End (TXE) Lock Retry Limit (-TXERetryLimit)....538
User index range (-userindexrangesize)....539
User table range (-usertablerangesize)....539
Database Client Notification (-usernotifytime)....540
Century Year Offset (-yy)....540

Chapter 24: PROMON Utility....543

PROMON utility....544
PROMON User Control option....545

PROMON Locking and Waiting statistics option....551
PROMON Block Access option....552
PROMON Record Locking Table option....554
PROMON Activity option....556
PROMON Shared Resources option....559
PROMON Database Status option....562
PROMON Shut Down Database option....564
PROMON Currently Connected Tenants option....564
PROMON R&D Advanced Options....565
R&D Status Displays....566
Database....566
Backup....568
Servers....569
Processes/Clients....570
Files....574
Lock Table....575
Buffer Cache....576
Logging Summary....577
BI Log....577
AI Log....578
Two-phase Commit....579
Startup Parameters....580
Shared Resources....582
Shared Memory Segments....583
AI Extents....583
Database Service Manager....585
Servers by broker....586
Client Database-Request Statement Cache....587
Schema Locks & Wait Queue....592
Broker Startup Parameters....592
R&D Activity Displays....592
Summary....594
Servers....596
Buffer Cache....597
Page Writers....598
BI Log....599
AI Log....600
Lock Table....601
I/O Operations By Type....602
I/O Operations by File....603
Space Allocation....604
Index....605
Record....605
Other....606
R&D Other Displays....607

Performance Indicators....607
I/O Operations by Process....609
Lock Requests by User....609
Checkpoints....610
I/O Operations by User by Table....612
I/O Operations by User by Index....612
Total Locks per User....613
R&D Administrative Functions....614
Check Active Transaction Status....614
Check Two-Phase Transactions....615
Resolve Limbo Transactions....615
Adjust Latch Options....616
Adjust Page Writer Options....616
Restricted Options....617
Server Options....617
Enable/disable block level consistency check....618
Shutdown Database....619
Disconnect a tenant's users....619
Adjust Governor options....620
Caches....620
Diagnostic Data Collection....621
R&D Adjust Monitor Options....622
PROMON 2PC Transactions Control option....623
PROMON Resolve 2PC Limbo Transactions option....625
PROMON 2PC Coordinator Information option....625
PROMON Resolve JTA Transactions option....626
PROMON Modify Defaults option....627

Chapter 25: PROUTIL Utility....629

PROUTIL utility syntax....632
PROUTIL 2PHASE BEGIN qualifier....638
PROUTIL 2PHASE COMMIT qualifier....638
PROUTIL 2PHASE END qualifier....639
PROUTIL 2PHASE MODIFY qualifier....639
PROUTIL 2PHASE RECOVER qualifier....640
PROUTIL AUDITARCHIVE qualifier....640
PROUTIL AUDITLOAD qualifier....643
PROUTIL AUDITRECONFIG qualifier....644
PROUTIL BIGROW qualifier....646
PROUTIL BULKLOAD qualifier....647
PROUTIL BUSY qualifier....648
PROUTIL CHANALYS qualifier....649
PROUTIL CODEPAGE-COMPILER qualifier....654
PROUTIL CONV1112 qualifier....654

PROUTIL CONVCHAR qualifier....656
PROUTIL CONVFILE qualifier....658
PROUTIL DATAMOVE qualifier....660
PROUTIL DBANALYS qualifier....661
Database analysis output....667
PROUTIL DBAUTHKEY qualifier....673
PROUTIL DBIPCS qualifier....674
PROUTIL DBRESTRICT qualifier....675
PROUTIL DEALLOCATE qualifier....677
PROUTIL DESCRIBE qualifier....678
PROUTIL DISABLEAUDITING qualifier....681
PROUTIL DISABLEAUTHGATEWAY qualifier....682
PROUTIL DISABLEB2 qualifier....683
PROUTIL DISABLECDC qualifier....683
PROUTIL DISABLEENCRYPTION qualifier....684
PROUTIL DISABLEJTA qualifier....685
PROUTIL DISABLEKEYEVENTS qualifier....686
PROUTIL DISABLEMULTITENANCY qualifier....686
PROUTIL DISABLEREADONLYPARTITIONS qualifier....687
PROUTIL DISABLETABLEPARTITIONING qualifier....687
PROUTIL DISPTOSSCREATELIMITS qualifier....688
PROUTIL DUMP qualifier....689
PROUTIL DUMPSPECIFIED qualifier....692
PROUTIL ENABLEAUDITING qualifier....695
PROUTIL ENABLEAUTHGATEWAY qualifier....696
PROUTIL ENABLEB2 qualifier....697
PROUTIL ENABLECDC qualifier....698
PROUTIL ENABLEENCRYPTION qualifier....699
PROUTIL ENABLEJTA qualifier....701
PROUTIL ENABLEKEYEVENTS qualifier....701
PROUTIL ENABLELARGEKEYS qualifier....702
PROUTIL ENABLEMULTITENANCY qualifier....702
PROUTIL ENABLEREADONLYPARTITIONS qualifier....703
PROUTIL ENABLESTOREDPROC qualifier....703
PROUTIL ENABLETABLEPARTITIONING qualifier....704
PROUTIL EPOLICY INFO qualifier....704
PROUTIL EPOLICY MANAGE qualifier....705
PROUTIL EPOLICY SCAN qualifier....707
PROUTIL EPOLICY VIEW qualifier....708
PROUTIL HOLDER qualifier....709
PROUTIL IDXACTIVATE qualifier....710
PROUTIL IDXANALYS qualifier....712
PROUTIL IDXBUILD qualifier....715
PROUTIL IDXCHECK qualifier....722
PROUTIL IDXCOMPACT qualifier....726

PROUTIL IDXDEACTIVATE 728
PROUTIL IDXFIX qualifier....730
PROUTIL IDXMOVE qualifier....734
PROUTIL INCREASETO qualifier....736
PROUTIL IOSTATS qualifier....738
PROUTIL LOAD qualifier....739
PROUTIL MTIDXBUILD qualifier....743
PROUTIL MVSCH qualifier....745
PROUTIL PARTITIONMANAGE ALTER qualifier....746
PROUTIL PARTITIONMANAGE COPY qualifier....748
PROUTIL PARTITIONMANAGE MERGE qualifier....750
PROUTIL PARTITIONMANAGE RENAME qualifier....751
PROUTIL PARTITIONMANAGE SPLIT qualifier....752
PROUTIL PARTITIONMANAGE TRUNCATE qualifier....753
PROUTIL PARTITIONMANAGE VIEW qualifier....755
PROUTIL RCODEKEY qualifier....759
PROUTIL SETAREACREATELIMIT qualifier....760
PROUTIL SETAREATOSSLIMIT qualifier....761
PROUTIL SETLOBCREATELIMIT qualifier....762
PROUTIL SETLOBTOSSLIMIT qualifier....763
PROUTIL SETTABLECREATELIMIT qualifier....764
PROUTIL SETTABLETOSSLIMIT qualifier....766
PROUTIL TABANALYS qualifier....767
PROUTIL TABLEMOVE qualifier....770
PROUTIL TPIDXBUILD qualifier....773
PROUTIL TRUNCATE AREA qualifier....775
PROUTIL TRUNCATE BI qualifier....776
PROUTIL UPDATESCHEMA qualifier....777
PROUTIL UPDATEVST qualifier....778
PROUTIL VIEWB2 qualifier....778
PROUTIL WBREAK-COMPILER qualifier....781
PROUTIL WORD-RULES qualifier....781
PROUTIL ZEROSTATS qualifier....782

Chapter 26: PROSTRCT Utility....785

PROSTRCT utility....786
PROSTRCT ADD qualifier....787
PROSTRCT ADDONLINE qualifier....788
PROSTRCT BUILDDB qualifier....789
PROSTRCT CLUSTER CLEAR qualifier....790
PROSTRCT CREATE qualifier....790
PROSTRCT LIST qualifier....792
PROSTRCT REMOVE qualifier....792
PROSTRCT REORDER AI qualifier....793

PROSTRCT REPAIR qualifier....794
PROSTRCT STATISTICS qualifier....794
PROSTRCT UNLOCK qualifier....795

Chapter 27: RFUTIL Utility....797

RFUTIL utility syntax....798
RFUTIL AIARCHIVER DISABLE qualifier....801
RFUTIL AIARCHIVER ENABLE qualifier....801
RFUTIL AIARCHIVER END qualifier....802
RFUTIL AIARCHIVER SETDIR qualifier....802
RFUTIL AIARCHIVER SETINTERVAL qualifier....803
RFUTIL AIARCHIVE EXTENT qualifier....803
RFUTIL AIARCHIVE NEXTEXTENT qualifier....804
RFUTIL AIMAGE AIOFF qualifier....805
RFUTIL AIMAGE BEGIN qualifier....806
RFUTIL AIMAGE END qualifier....806
RFUTIL AIMAGE EXTENT EMPTY qualifier....807
RFUTIL AIMAGE EXTENT FULL qualifier....808
RFUTIL AIMAGE EXTENT LIST qualifier....808
RFUTIL AIMAGE EXTRACT qualifier....809
RFUTIL AIMAGE NEW qualifier....809
RFUTIL AIMAGE QUERY qualifier....810
RFUTIL AIMAGE SCAN qualifier....811
RFUTIL AIMAGE TRUNCATE qualifier....812
RFUTIL AIVERIFY PARTIAL qualifier....813
RFUTIL AIVERIFY FULL qualifier....814
RFUTIL MARK BACKEDUP qualifier....815
RFUTIL ROLL FORWARD qualifier....815
RFUTIL ROLL FORWARD OPLOCK qualifier....816
RFUTIL ROLL FORWARD RETRY qualifier....817
RFUTIL ROLL OPUNLOCK qualifier....819
RFUTIL SEQUENCE qualifier....819

Chapter 28: Other Database Administration Utilities....821

Parameters for authentication....822
DBMAN utility....822
DBTOOL utility....825
Running DBTOOL....828
DSRUTIL utility....830
PROADSV utility....830
PROBKUP utility....832
PROCLUSTER utility....836
PROCOPY utility....839

PRODB utility....841
PRODEL utility....842
PROLOG utility....843
PROREST utility....844

Chapter 29: SQL Utilities....849

SQLDUMP utility....849
SQLLOAD utility....853
SQLSCHEMA utility....857

Chapter 30: Virtual System Tables....863

Update access to virtual system tables....864
Accessing VSTs with ABL or SQL....864
Modifiable VSTs on Replication Targets....864
Virtual system table summaries....867
After-image log activity (_ActAILog)....867
Before-image log activity (_ActBILog)....868
Buffer activity (_ActBuffer)....869
Index activity (_ActIndex)....871
Input/output activity (_ActIOFile)....871
Input/output type activity (_ActIOType)....872
Lock table activity (_ActLock)....872
Other activity (_ActOther)....873
Page writer activity (_ActPWs)....874
Record activity (_ActRecord)....875
Server activity (_ActServer)....875
Space allocation activity (_ActSpace)....876
Summary activity (_ActSummary)....877
Area status (_AreaStatus)....878
Area threshold (_AreaThreshold)....879
Block (_Block)....879
Buffer status (_BuffStatus)....880
Cache (_Cache)....881
Checkpoint (_Checkpoint)....881
Code Features (_Code-Feature)....882
Database connection (_Connect)....883
Database features (_Database-Feature)....887
Database startup parameters file status (_DbParams)....888
Database status (_DbStatus)....888
Database file status (_Filelist)....890
Index statistics (_IndexStat)....890
Latch statistics (_Latch)....891
License management (_License)....892

Lock table status (_Lock)....892
Lock request (_LockReq)....893
Logging (_Logging)....894
Master block (_MstrBlk)....896
User connection (_MyConnection)....897
Resource queue statistics (_Resrc)....897
Segments (_Segments)....898
Servers (_Servers)....898
SQL query plan (_SQL_Qplan)....899
Table statistics (_TableStat)....899
Transaction (_Trans)....900
Transaction end lock statistics (_TxeLock)....900
User index activity (_UserIndexStat)....901
Database input/output (_UserIO)....902
Record locking table (_UserLock)....902
User status (_UserStatus)....903
User table activity (_UserTableStat)....908

Index....911

Preface

For details, see the following topics:

- Purpose
- Audience
- Organization
- Using ABL documentation
- Typographical conventions
- Examples of syntax descriptions
- OpenEdge messages

Purpose

This manual describes OpenEdge® RDBMS administration concepts, procedures, and utilities. The procedures allow you to create and maintain your OpenEdge databases and manage their performance. This manual assumes that you are familiar with the OpenEdge RDBMS planning concepts discussed in *OpenEdge Getting Started: Database Essentials*.

Audience

This manual is designed as a guide and reference for OpenEdge Database Administrators.

Organization

This book is organized into four complementary parts:

- Part I, "Database basics," describes the basic commands for creating and deleting, and starting up and shutting down databases, along with detailing database limits:

  Creating and Deleting Databases on page 41

  Describes how to create and delete OpenEdge databases.

  OpenEdge RDBMS Limits on page 59

  Catalogs limits of the OpenEdge RDBMS, including all aspects of database size, operating system limits, naming conventions, and data types.

  Starting Up and Shutting Down on page 73

  Describes the commands required to start up and shut down an OpenEdge database.

- Part II, "Protecting your data," describes the procedures a database administrator uses to protect a database in a flexible business environment. Each chapter discusses a particular administrative activity:

  Backup Strategies on page 89

  Discusses various approaches to backing up your database.

  Backing Up a Database on page 97

  Describes the mechanics of backing up your database with the PROBKUP utility.

  Recovering a Database on page 117

  Examines recovery strategies and how to use the PROREST utility to restore an OpenEdge database.

  After-imaging on page 139

  Presents after-imaging and how to use it for data recovery. Also, describes how to implement after-imaging with after-image extents.

  Maintaining Security on page 169

  Describes how to implement database security, including assigning user IDs and designating database administrators.

  Auditing on page 179

  Introduces auditing. How to enable and disable auditing on your database, and what can be audited is discussed.

  Transparent Data Encryption on page 199

  Introduces Transparent Data Encryption. Describes how to enable and disable transparent data encryption, create encryption policies, and how to maintain a database enabled for transparent data encryption.

  Replicating Data on page 229

  Examines replication schemes and how to implement log-based replication.

  Failover Clusters on page 235

  Explains how to configure and manage a cluster-enabled database.

  Distributed Transaction Processing on page 255

  Explains distributed transaction processing, and discusses support for two-phase commit and the Java Transaction API (JTA).

- Part III, "Maintaining and monitoring your database," describes the procedures and tools a database administrator employs to keep a database functioning efficiently:

  Managing Performance on page 275

  Discusses how to monitor tune database performance.

  Maintaining Database Structure on page 325

  Describes methods to manage the database structure and alter it as necessary to improve storage and performance.

  Multi-tenant database on page 347

  Describes the architecture and features of a multi-tenant database.

  Table-partitioned database on page 371

  Describes the architecture and process of table partitioning.

  Change Data Capture on page 389

  Describes Change Data Capture in a database.

  Describes the architecture and features of a table-partitioned database.

  Dumping and Loading on page 411

  Explains how to dump and load databases, including tables, indexes, and sequences.

  Logged Data on page 439

  Examines the process of logging significant database events.

- Part IV, "Reference," describes the OpenEdge RDBMS commands, startup parameters, utilities, and system tables. Refer to the chapters in Part IV when you need to access specific descriptive information, such as the syntax of an administration utility:

  Startup and Shutdown Commands on page 461

  Catalogs the OpenEdge RDBMS commands for starting up and shutting down database sessions and processes.

  Database Startup Parameters on page 473

  Lists and details the OpenEdge RDBMS startup parameters.

  PROMON Utility on page 543

  Details the PROMON Utility used for monitoring your database.

  PROUTIL Utility on page 629

  Details the PROUTIL Utility used for maintaining your database.

  PROSTRCT Utility on page 785

  Details the PROSTRCT Utility used for creating and updating the physical structure of your database.

  RFUTIL Utility on page 797

  Details the RFUTIL Utility used for managing after imaging.

  Other Database Administration Utilities on page 821

  Details other database utilities including PROBKUP, PROREST, PROCOPY, PRODEL, and PROLOG.

  SQL Utilities on page 849

  Details the utilities used for maintaining your database for use with SQL.

  Virtual System Tables on page 863

Describes the Virtual System Tables that allow ABL and SQL applications to examine the status of a database and monitor its performance.

Using ABL documentation

OpenEdge provides a special purpose programming language for building business applications. In the documentation, the formal name for this language is *ABL (Advanced Business Language)*. With few exceptions, all keywords of the language appear in all `UPPERCASE`, using a font that is appropriate to the context. All other alphabetic language content appears in mixed case.

For the latest documentation updates go to https://docs.progress.com.

References to ABL compiler and run-time features

ABL is both a compiled and an interpreted language that executes in a run-time engine. The documentation refers to this run-time engine as the *ABL Virtual Machine (AVM)*. When the documentation refers to ABL source code compilation, it specifies *ABL* or *the compiler* as the actor that manages compile-time features of the language. When the documentation refers to run-time behavior in an executing ABL program, it specifies *the AVM* as the actor that manages the specified run-time behavior in the program.

For example, these sentences refer to the ABL compiler's allowance for parameter passing and the AVM's possible response to that parameter passing at run time: "ABL allows you to pass a dynamic temp-table handle as a static temp-table parameter of a method. However, if at run time the passed dynamic temp-table schema does not match the schema of the static temp-table parameter, the AVM raises an error." The following sentence refers to run-time actions that the AVM can perform using a particular ABL feature: "The ABL socket object handle allows the AVM to connect with other ABL and non-ABL sessions using TCP/IP sockets."

References to ABL data types

ABL provides built-in data types, built-in class data types, and user-defined class data types. References to built-in data types follow these rules:

- Like most other keywords, references to specific built-in data types appear in all `UPPERCASE`, using a font that is appropriate to the context. No uppercase reference ever includes or implies any data type other than itself.
- Wherever *integer* appears, this is a reference to the `INTEGER` or `INT64` data type.
- Wherever *character* appears, this is a reference to the `CHARACTER`, `LONGCHAR`, or `CLOB` data type.
- Wherever *decimal* appears, this is a reference to the `DECIMAL` data type.
- Wherever *numeric* appears, this is a reference to the `INTEGER`, `INT64`, or `DECIMAL` data type.

References to built-in class data types appear in mixed case with initial caps, for example, `Progress.Lang.Object`. References to user-defined class data types appear in mixed case, as specified for a given application example.

Typographical conventions

This documentation uses the following typographical and syntax conventions:

Convention	Description
Bold	Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.
Italic	Italic typeface indicates the title of a document, or signifies new terms.
SMALL, BOLD CAPITAL LETTERS	Small, bold capital letters indicate OpenEdge key functions and generic keyboard keys; for example, **GET** and **CTRL**.
KEY1+KEY2	A plus sign between key names indicates a **simultaneous** key sequence: you press and hold down the first key while pressing the second key. For example, **CTRL+X**.
KEY1 KEY2	A space between key names indicates a **sequential** key sequence: you press and release the first key, then press another key. For example, **ESCAPE H**.
Syntax:	
`Fixed width`	A fixed-width font is used in syntax, code examples, system output, and file names.
`Fixed-width italics`	Fixed-width italics indicate variables in syntax.
`Fixed-width bold`	Fixed-width bold italic indicates variables in syntax with special emphasis.
`UPPERCASE fixed width`	ABL keywords in syntax and code examples are almost always shown in upper case. Although shown in uppercase, you can type ABL keywords in either uppercase or lowercase in a procedure or class.
Period (.) or colon (:)	All statements except `DO`, `FOR`, `FUNCTION`, `PROCEDURE`, and `REPEAT` end with a period. `DO`, `FOR`, `FUNCTION`, `PROCEDURE`, and `REPEAT` statements can end with either a period or a colon.
[ ]	Large brackets indicate the items within them are optional.
[ ]	Small brackets are part of ABL.
{ }	Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams.
{ }	Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure.

Convention	Description
|	A vertical bar indicates a choice.
. . .	Ellipses indicate repetition: you can choose one or more of the preceding items.

Examples of syntax descriptions

In this example, ACCUM is a keyword, and *aggregate* and *expression* are variables:

Syntax

```
ACCUM aggregate expression
```

FOR is one of the statements that can end with either a period or a colon, as in this example:

```
FOR EACH Customer NO-LOCK:
  DISPLAY Customer.Name.
END.
```

In this example, STREAM *stream*, UNLESS-HIDDEN, and NO-ERROR are optional:

Syntax

```
DISPLAY [ STREAM stream ] [ UNLESS-HIDDEN ] [ NO-ERROR ]
```

In this example, the outer (small) brackets are part of the language, and the inner (large) brackets denote an optional item:

Syntax

```
INITIAL [ constant [ , constant ] ]
```

A called external procedure must use braces when referencing compile-time arguments passed by a calling procedure, as shown in this example:

Syntax

```
{ &argument-name }
```

In this example, EACH, FIRST, and LAST are optional, but you can choose only one of them:

Syntax

```
PRESELECT [ EACH | FIRST | LAST ] record-phrase
```

In this example, you must include two expressions, and optionally you can include more. Multiple expressions are separated by commas:

Syntax

```
MAXIMUM ( expression , expression [ , expression ] ... )
```

In this example, you must specify MESSAGE and at least one *expression* or SKIP [(*n*)], and any number of additional *expression* or SKIP [(*n*)] is allowed:

Syntax

```
MESSAGE { expression | SKIP [ ( n ) ] } ...
```

In this example, you must specify {*include-file*, then optionally any number of *argument* or &*argument-name* = "*argument-value*", and then terminate with }:

Syntax

```
{ include-file
    [ argument | &argument-name = "argument-value" ] ... }
```

Long syntax descriptions split across lines

Some syntax descriptions are too long to fit on one line. When syntax descriptions are split across multiple lines, groups of optional and groups of required items are kept together in the required order.

In this example, WITH is followed by six optional items:

Syntax

```
WITH [ ACCUM max-length ] [ expression DOWN ]
  [ CENTERED ] [ n COLUMNS ] [ SIDE-LABELS ]
  [ STREAM-IO ]
```

Complex syntax descriptions with both required and optional elements

Some syntax descriptions are too complex to distinguish required and optional elements by bracketing only the optional elements. For such syntax, the descriptions include both braces (for required elements) and brackets (for optional elements).

In this example, `ASSIGN` requires either one or more *`field`* entries or one *`record`*. Options available with *`field`* or *`record`* are grouped with braces and brackets:

Syntax

```
ASSIGN   { [ FRAME frame ] { field [ = expression ] }
           [ WHEN expression ] } . . .
       | { record [ EXCEPT field . . . ] }
```

OpenEdge messages

OpenEdge displays several types of messages to inform you of routine and unusual occurrences:

- **Execution messages** inform you of errors encountered while OpenEdge is running a procedure; for example, if OpenEdge cannot find a record with a specified index field value.
- **Compile messages** inform you of errors found while OpenEdge is reading and analyzing a procedure before running it; for example, if a procedure references a table name that is not defined in the database.
- **Startup messages** inform you of unusual conditions detected while OpenEdge is getting ready to execute; for example, if you entered an invalid startup parameter.

After displaying a message, OpenEdge proceeds in one of several ways:

- Continues execution, subject to the error-processing actions that you specify or that are assumed as part of the procedure. This is the most common action taken after execution messages.
- Returns to the Procedure Editor, so you can correct an error in a procedure. This is the usual action taken after compiler messages.
- Halts processing of a procedure and returns immediately to the Procedure Editor. This does not happen often.
- Terminates the current session.

OpenEdge messages end with a message number in parentheses. In this example, the message number is `200`:

```
** Unknown table name table. (200)
```

If you encounter an error that terminates OpenEdge, note the message number before restarting.

Obtaining more information about OpenEdge messages

In Windows platforms, use OpenEdge online help to obtain more information about OpenEdge messages. Many OpenEdge tools include the following Help menu options to provide information about messages:

- Choose **Help** > **Recent Messages** to display detailed descriptions of the most recent OpenEdge message and all other messages returned in the current session.
- Choose **Help** > **Messages** and then type the message number to display a description of a specific OpenEdge message.
- In the Procedure Editor, press the **HELP** key or **F1**.

On UNIX platforms, use the OpenEdge `pro` command to start a single-user mode character OpenEdge client session and view a brief description of a message by providing its number.

To use the pro command to obtain a message description by message number:

1. Start the Procedure Editor:

```
OpenEdge-install-dir/bin/pro
```

2. Press **F3** to access the menu bar, then choose **Help** > **Messages**.
3. Type the message number and press **ENTER**. Details about that message number appear.
4. Press **F4** to close the message, press **F3** to access the Procedure Editor menu, and choose **File** > **Exit**.

I

Database Basics

For details, see the following topics:

- Creating and Deleting Databases
- OpenEdge RDBMS Limits
- Starting Up and Shutting Down

1

Creating and Deleting Databases

This chapter describes the methods to create and delete an OpenEdge database.

For details, see the following topics:

- Ways to create an OpenEdge database
- Database creation and large file support
- Creating a database with PROSTRCT CREATE
- Creating a database with the PRODB utility
- PRODB maintains pathname convention
- Creating a database with a Data tool
- Converting an OpenEdge Release 11 Database to OpenEdge Release 12
- Using the Schema Mover
- Converting an OpenEdge Release 10 database to OpenEdge Release 12
- Database conversion utilities
- Copying a database
- Deleting a database

Ways to create an OpenEdge database

There are several ways to create an OpenEdge database. You can use:

- A structure description file with the PROSTRCT CREATE utility
- The PRODB utility
- The Data Dictionary if you are using a graphical interface or a character interface
- The Data Administration if you are using a graphical interface
- The PROCOPY utility
- The PROREST utility
- An OpenEdge Release 10 database, and convert it to OpenEdge Release 11 format

When you create a database, you can create either of the following:

- A new but empty database
- A copy of an existing database

Note: Do not create your database in the OpenEdge Install directory or in any subdirectory of the Install directory. Databases residing in these directories cannot be opened.

Database creation and large file support

Large file support for databases created, copied, or opened using OpenEdge Release 12.0 and later will differ based on license level. This change will affect databases created or copied using the following commands and utilities:

- PROSTRCT CREATE
- PROCOPY
- PRODB
- CONV1112
- PROREST

For systems running with Enterprise license, large file support will be enabled by default.

Opening a database on systems running with Enterprise license will enable large file support if it is not already enabled. The following message will be written to the database .lg file upon enabling large file support:

```
Database with large file support disabled is being opened under
Enterprise license. Large file support will be enabled. (19400)
```

Large file support is not available for systems running with sub-Enterprise license (Workgroup and Personal) and is disabled by default.

Opening a database on systems running with sub-Enterprise license will disable large file support if it is enabled. The following message will be written to the database .lg file upon disabling large file support:

```
Database with large file support enabled is being opened under
sub-Enterprise license. Large file support will be disabled.
(19399)
```

In the event existing file extents exceed 2GB when opening a database running with sub-Enterprise license, the database will fail to open and the following message will appear on screen and in the .lg file:

```
Database having existing file extents greater than the 2GB large
file size limit cannot be opened under a sub-Enterprise license.
(19401)
```

Large file support may be enabled or disabled depending on the license when the database is opened in a non-read only mode.

Note: Although large file support is enabled in an OpenEdge database, it is still limited by the operating system support.

Creating a database with PROSTRCT CREATE

To create an OpenEdge database using PROSTRCT CREATE you must:

- Create a structure description (`.st`) file to define storage areas and extents
- Use PROSTRCT CREATE to create database areas and extents
- Add schema to the void database

Notes

- Changes to large file support in OpenEdge Release 12.0 will impact databases created using PROSTRCT CREATE. See Database creation and large file support on page 42 for more information.

Creating a structure description file

The structure description file is a text file you prepare that defines the database structure. It contains all of the information required by the PROSTRCT CREATE utility to create a database control area and the database extents.

Use a text editor to create the structure description file. The name you give to the structure description file is usually the name of the database you define, with a `.st` extension.

The structure description file contains one or more lines of text that provide information about each storage area of the database. Each line of text is composed of *tokens*, which are text strings made up of alphanumeric characters that describe the following characteristics:

- The storage area type
- The area name
- The area number
- Optionally, the records per block
- Optionally, the number of blocks per cluster (for Type II data storage areas)
- The extent pathname
- The extent type
- The extent size

The following grammar rules how the 8 tokens are combined to form line in a structure description file:

```
line     = comment | CR | type path [sizeinfo]

comment  = * | : | #

CR       = blank line

type     = a | b | d | t [areainfo]
   areainfo = "areaname" [:areanum] [,recsPerBlock] [;blksPerCluster]
      areaname       = string
      areanum        = numeric value
      recsPerBlock   = numeric value
      blksPerCluster = 1 | 8 | 64 | 512

path     = string

sizeinfo = extentType size
     extentType     = f | v
     size           = numeric value > 32
```

Note: You can comment a `.st` file and use blank lines. Precede comments with a pound sign (#), colon (:), or asterisk (*).

The table below explains the value of each of the 8 tokens in a structure description file.

Table 1: ST file tokens

Token	Description
type	Indicates the type of storage area. Possible values are: • **a** — After-image area • **b** — Before-image area • **d** — Schema and application data areas • **t** — Transaction log area
areaname	Name of the storage area.
areanum	Number of the storage area. **Note:** If you do not include the area number token, PROSTRCT will assign the area numbers for you. However, if you choose to specify a number for one area in your `.st`, you must specify area numbers for all the areas or PROSTRCT will return an error.

Token	Description
recsPerBlock	Number of database records in each database block. Valid values are 1, 2, 4, 8, 16, 32, 64, 128, and 256. **Note:** • This token only applies to areas 7 through 32,000. • For the schema area, records per block is fixed at 32 for 1K, 2K, and 4K database block sizes, The records per block is 64 for an 8K database block size.
blcksPerCluster	Number of database blocks in each cluster. Possible values are: 1, 8, 64, or 512. **Note:** • If you leave this value blank, or specify 1, the data area will be Type I. All other values are valid for Type II data areas only. • This token only applies to areas 7 through 32,000.
path	Absolute or relative pathname of each extent.
extentType	Indicates whether the extent is fixed (`f`) or variable (`v`). If the extent type token is not specified, the extent is variable.
size	Size of an extent in kilobytes. This value must be a multiple of 16 times your database block size.

Extent pathnames and naming conventions

The PROSTRCT CREATE utility is designed to allow the end user to specify the minimum amount of information necessary to create a database. Only the area type and extent location must be specified. A suggested convention is that you supply an area name for data storage areas (`d`). A specific filename or file extension need not be provided. If you specify a pathname, it must represent a standard operating system file. The only pathname restrictions are those that might be imposed by your operating system.

The PROSTRCT CREATE utility will generate filename and file extensions for all database files according to the following naming convention:

- The control area (`.db`) and the log file (`.lg`) are placed in the directory specified by the command line dbname parameter.
- If a relative pathname is provided, including using common dot (.) notation, the relative pathname will be expanded to an absolute pathname. Relative paths begin in your current working directory.
- For before-image extents, the filename is the database name with a `.bn` extension, where *n* represents the order in which the extents were created, beginning with `1`.
- For after-image extents, the filename is the database name with a `.an` extension, where *n* represents the order in which the extents were created, beginning with `1`.

- For transaction log extents, the filename is the database name with a `.tn` extension, where *n* represents the order in which the extents were created, beginning with `1`.
- For schema area extents, the filename is the database name with a `.dn` extension, where *n* represents the order in which the extents were created and will be used.
- For application data area extents, the filename is the database name followed by an underscore and the area number (for example, `customer_7.d1`). The area number is a unique identifier that differentiates between different areas. The application data area extent filenames also have a `.dn` extension, where *n* represents the order in which the extents were created and will be used.

Note: In a structure description (`.st`) file, to specify a pathname that contains spaces (such as `\usr1\misc data`), precede the pathname with an exclamation point (`!`) and wrap the pathname in quotation marks (`" "`). For example, `!"\usr1\misc data"`.

Rules for creating storage areas and extents

When you are defining storage areas and extents in order to create a new database:

- The minimum information required in a `.st` file is one schema area extent definition statement and one primary recovery (BI) area extent definition statement.
- The minimum information needed to specify any extent is the storage area type and extent pathname. For example:

```
# Primary Recovery Area
b .
# Schema Area
d .
```

  If you do not define a primary recovery extent path in the `.st` file, the PROSTRCT CREATE utility generates an error.

- You cannot use any of the reserved storage area names as application data storage area names.

Extent length

You can specify a fixed-length or a variable-length extent:

- **Fixed-length** — When you create a fixed-length extent, its blocks are preallocated and preformatted specifically for the database. If you want the extent to be fixed length, the extent type token of the extent description line is `f`. This token must be lowercase. If the extent is fixed length, use the extent size token to indicate its length in kilobytes.

  The size of the extent, in kilobytes, must be a multiple of (16 * *database-blocksize*). If you specify a size that is not a multiple of this, PROSTRCT CREATE displays a warning message and rounds the size up to the next multiple of (16 * *database-blocksize*). The minimum length for a fixed-length file is 32K, and the maximum length of a file depends on the size of the file system and the physical volume containing the extent.

  The table below shows how the extent size changes based on the database block size.

Table 2: Calculating extent size

Database block size	Formula	Extent size
1	2*(16 * 1)	32K
	3*(16 * 1)	48K
2	1*(16 * 2)	32K
	2*(16 * 2)	64K
	3*(16 * 2)	96K
4	1*(16 * 4)	64K
	2*(16 * 4)	128K
	3*(16 * 4)	192K
8	1*(16 * 8)	128K
	2*(16 * 8)	256K
	3*(16 * 8)	384K

- Variable-length — Typically, you use a variable-length extent as an overflow file when all fixed-length extents have been exhausted. For DB and BI extents, you can define one variable-length extent for each area, and it must be the last extent in the area. There is no limit to the number of variable-length AI extents you can define. While you indicate a variable-length extent by leaving out the extent size in the `.st` file entry line, you can also specify the maximum size to which the extent can grow by indicating the "`v`" extent type and a size in kilobytes. The initial allocation for a variable-length extent is 32K or the size of the cluster (whichever is larger). Calculate the cluster size by multiplying your block size times your blocks per cluster value.

Note: Regardless of whether the extent is fixed- or variable-length, if it is in a data area with clusters its size must be large enough to hold an entire cluster, or PROSTRCT generates and error.

Example structure description file

The example that follows shows a `.st` file named `sports2000.st` that defines a database with:

- One primary recovery area.
- One schema area.
- Three after-image areas each with a fixed-length extent.
- One transaction log area with a fixed-length extent used with two-phase commit.
- Six application data areas each with one fixed- and one variable-length extent. The area names for the six application data areas are: Employee, Inventory, Cust_Data, Cust_Index, Order, and Misc. Note that the Cust_Data, Cust_Index, and Order areas have cluster sizes assigned to them, and are therefore Type II

storage areas. The area numbers are not specified. The areas will be numbered sequentially starting at 7. Blocks per cluster is not specified, creating Type I data areas.

```
Sample Structure Description File: sports2000.st
#
# The following defines the Primary Recovery Area consisting of one
# variable length extent. It resides in the /usr2/bi directory:
#
b /usr2/bi
#
# The following defines the Schema Area consisting of one variable length # extent,
 residing in the current working directory:
#
d "Schema Area" .
#
# The following defines three fixed length After Image Areas each equal to
# 1 MB in size:
#
a /usr3/ai f 1024
a /usr3/ai f 1024
a !"/usr3/ai data" f 1024
#
# The following defines a Transaction Log Area equal to 4 MB in size and
# residing in the current working directory. This storage area is used
# for 2 phase commit:
#
t . f 4096
#
# The following defines six Application Data Areas each with one fixed
# length extent equal to 1 MB in size and 1 variable length extent:
#
d "Employee",32 /usr1/emp f 1024
d "Employee",32 /usr1/emp
#
d "Inventory",32 /usr1/inv f 1024
d "Inventory",32 /usr1/inv
#
d "Cust_Data",32;64 /usr1/cust f 1024
d "Cust_Data",32;64 /usr1/cust
#d "Cust_Index",32;8 /usr1/cust f 1024
d "Cust_Index",32;8 /usr1/cust
#
d "Order",32;64 /usr1/ord f 1024
d "Order",32;64 /usr1/ord
#
d "Misc",32 !"/usr1/misc data" f 1024
d "Misc",32 !"/usr1/misc data"
#
# Note that the directory pathname for the "Misc" application data area
# contains spaces, and to recognize that the pathname is specified with
# an ! (exclamation point) and " " (quotation marks).
```

Example structure description file for large files

When creating a new database, large file processing is enabled if the `.st` file specifies a fixed-length extent size or a maximum size for a variable-length extent that is greater than 2 GB. Large file processing requires an Enterprise database license and is enabled by default. Large file processing is not available for sub-Enterprise license and is disabled by default. The following example shows the `.st` file of a database with large file processing enabled:

```
# Sample Structure Description File: largedb.st
#
# largedb.st to create database largedb with large file processing enabled.
#
# A fixed length bi file of 1GB and a variable length bi file with a maximum # size of
 4GB.
#
b tests/largedb.b1 f 1048576
b tests/largedb.b2 v 4194304
#
# SCHEMA AREA with a fixed length file of 3GB and a variable length file with # a maximum
 size of 3GB.
#
d "Schema Area":6,64 tests/largedb.d1 f 3145728
d "Schema Area":6,64 tests/largedb.d2 v 3145728
#
# TABLE AREA with a fixed length file of just over 2GB and a variable length # file
with a maximum size of 1TB.
#
d "Table Area":7,64 tests/largedb_7.d1 f 2097280
d "Table Area":7,64 tests/largedb_7.d2
#
# A fixed length ai file of 2GB and a variable length file with a maximum size # of
1TB.
a tests/largedb.a1 f 2097152
a tests/largedb.a2
#
```

Create a database structure extent

Use the PROSTRCT CREATE utility to create the physical database files for the database. The PROSTRCT CREATE syntax is:

```
prostrct create db-name [structure-description-file]
   [-blocksize blocksize]
   [-validate]
```

In the syntax block, *structure-description-file* represents the name of the `.st` file. If you do not specify the name of the `.st` file, PROSTRCT uses *db-name*`.st`. The database blocksize in kilobytes is represented by *blocksize*. Specify 1024, 2048, 4096, or 8192. If you specify `-validate`, PROSTRCT will check that the contents of your structure description file are accurate without creating the database.

To create a database named `sports2000` from the `sports2000.st` structure description file using PROSTRCT CREATE:

1. First verify that your structure description file is accurate.

Use the -validate option to PROSTRCT CREATE as follows:

```
prostrct create sports2000 sports2000.st -blocksize 4096 -validate
```

If there are no errors in your structure description file, PROSTRCT returns a status similar to the following:

```
The structure file format is valid. (12619)
Device: /usr1/dbs/, KBytes needed: 3488, KBytes available: 483876225
(12616)
There is sufficient free space to initialize the defined extents. (12618)
```

2. Second, create the database.

   Use the PROSTRCT CREATE command as follows:

```
prostrct create sports2000 sports2000.st -blocksize 4096
```

Adding schema to a void database

When you use the PROSTRCT CREATE utility on a .st file, the resulting database files are referred to as a *void database*. A void database does not contain any OpenEdge RDBMS metaschema. The void database consists of the .db extent and whatever .bi, .ai, .tl, and .d*n* extents you defined in the .st file. You must add metaschema information to a void database. The OpenEdge RDBMS provides an empty database in the size of each supported database block size for this purpose.

To add metaschema information to a void database:

1. Copy the system tables from an empty OpenEdge database into the void database you created with PROSTRCT CREATE.

   Use the PROCOPY utility as shown:

```
procopy OpenEdge-install-dir/emptyn db-name
```

   Where empty*n* is the source database and *db-name* is the target database. *n* indicates the block size of the OpenEdge-supplied empty.db. See PRODB utility on page 841 for more information about the empty and empty*n* databases.

2. Verify that you have the correct database files in the correct locations

   Use the PROSTRCT LIST utility as shown:

```
prostrct list db-name
```

Caution: If you do not specify an output file when using PROSTRCT LIST, your existing file will be overwritten to reflect the current structure.

3. Use the Data Dictionary to load the existing user table definitions (`.df` file) into your database.

Caution: The empty `.db` file and the database you want copied to it must have the same block size. Similarly, if the blocks per cluster token is used with any data areas, the data areas of the empty `.db` file and those of the database you want copied to it must have the same blocks per cluster value.

Creating a database with the PRODB utility

If the target database does not have a `.st` file, the PRODB utility creates a new database from a specified source database. PRODB creates a new database using the structure of the source database and places all of the extents in the current working directory. You can use PRODB to make a copy of any of the demonstration or empty OpenEdge databases.

Note: When using PRODB to create a copy of a database, all the files of the database copy will reside in the same directory, unless a `.st` file already exists for the target database.

See PRODB utility on page 841 for more information about PRODB.

Examples

The following examples illustrate how to create databases with PRODB:

- To create an empty database called `mysample` from a copy of the default empty database, enter the following:

```
prodb mysample empty
```

- To create a new database called `mysports2000` from a copy of the sports2000 database, enter the following:

```
prodb mysports2000 sports2000
```

- To create a new database called `pastinfo` from a copy of an existing database named `currentinfo`, enter the following:

```
prodb pastinfo currentinfo
```

PRODB does not copy the external triggers associated with the database you are copying.

Notes

- Changes to large file support in OpenEdge Release 12.0 will impact databases created using PRODB. See Database creation and large file support on page 42 for more information.

PRODB maintains pathname convention

When you use the PRODB utility, the target database you create maintains the same pathname convention, relative or absolute, as the source database. For example, if you use PRODB to create a database and name it `example1`, and use a relative path database such as `sports2000` as the source database, PRODB maintains the pathname convention of `sports2000` (the source database), and `example1` (the target database) becomes a relative path database. Conversely, if the source database is an absolute path database, the target database you create with PRODB will also be an absolute path database. Use PROSTRCT LIST to verify whether the pathname is relative or absolute. For example:

```
prodb example1 sports2000
prostrct list example1
```

In the following sample output of the PROSTRCT LIST utility, note dot (.) notation of the relative pathname of the database, `example1.db`:

```
Area Name: Control Area, Type 6, BlockSize 4096, Extents 1, Records/Block32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 32 KByte, Name: ./example1.db

Area Name: Primary Recovery Area, Type 3, BlockSize 8192, Extents 1
   Ext # 1, Type VARIABLE, Size 2176, Name: ./example1.b1

Area Name: Schema Area, Type 6, BlockSize 4096, Extents 1, Records/Block 32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 960 KByte, Name: ./example1.d1

Area Name: Info, Type 6, BlockSize 4096, Extents 1, Records/Block 32,
Cluster Size 1
   Ext #1, Type VARIABLE, Size 192 KByte, Name: ./example1_7.d1

Area Name: Order, Type 6, BlockSize 1024, Extents 1, Records/Block 32,
Cluster Size 1
   Ext # 1, Type FIXED   , Size 1280, Name: ./example1_8.d1
```

Note: See *OpenEdge Getting Started: Database Essentials* for an explanation of relative- and absolute-pathnames. See Maintaining Database Structure on page 325 for information about changing pathname conventions when adding storage areas and extents to a structure description file.

Creating a database with a Data tool

You have several option for creating and maintaining OpenEdge databases:

- **Data Dictionary** — The Data Dictionary allows you to create OpenEdge databases and maintain schema in a graphical or character interface. You can define and update tables, fields, indexes, and sequences.

- **Data Administration** — The Data Administration provides a graphical interface for creating and maintaining OpenEdge databases. Available administration tasks include dumping and loading data and schema, along with tasks related to reporting, security, and DataServers.
- **Database Administration Console** -- As part of OpenEdge Explorer and OpenEdge Management, the Database Administration Console provides a browser-based interface for maintaining database schema, particularly multi-tenant schema.

For more information on:

- Graphical Data Dictionary, see the online Help
- Character Data Dictionary, see *OpenEdge Development: Basic Database Tools*
- Data Administration, see the online Help
- Database Administration Console, see *OpenEdge Management and OpenEdge Explorer: Configuration* and the online Help

Converting an OpenEdge Release 11 Database to OpenEdge Release 12

The PROUTIL CONV1112 utility converts the schema of an OpenEdge Release 11 database to OpenEdge Release 12.

To convert the schema of an OpenEdge Release 11 database:

1. Back up the database using OpenEdge Release 11.

   Caution: There is always a chance that your schema could become corrupt during conversion. If the conversion fails, your database cannot be recovered. If this happens, you must restore the backup copy of your database and begin the conversion again.

2. Verify that the database is not in use.
3. If your OpenEdge Release 11 database is enabled for Transparent Database Encryption (TDE) or has area, table, or index data objects currently encrypted with RC4, you must complete the following steps before running the PROUTIL CONV1112 utility. If your database is not TDE-enabled, or is not using RC4 for any of its areas, tables, or indexes, you may skip to step 4.
   a) Modify your OpenEdge Release 11 database to change the TDE policy for each RC4-encrypted table or index to use an encryption algorithm that is supported in OpenEdge 12, such as AES-128. For details, see Changing the cipher of an encryption policy with PROUTIL EPOLICY in *Introduction to Security and Auditing*.
   b) Update the encrypted data that is stored in the database. For an example, see EPOLICY MANAGE UPDATE example in *Introduction to Security and Auditing* (an example of updating is described in steps 4 and 5).
   c) Make sure that the updated tables and indexes show that all of the stored data is encrypted with the new algorithm.
4. Truncate your before-image file with Release 11 PROUTIL TRUNCATE BI. You cannot truncate the before-image file of your Release 11 database with Release 12.
5. Disable after-imaging, if enabled, with Release 11 RFUTIL AIMAGE END.

If you have OpenEdge Replication enabled, you must disable it before you can disable after-imaging. For information on Replication, see *Use Database Replication*.

6. Disable two-phase commit, if enabled, with Release 11 PROUTIL 2PHASE END.
7. Verify your Release 11 backup, and then install OpenEdge Release 12, if you have not already done so, following the instructions in *Install OpenEdge*.
8. Run PROUTIL CONV1112 to convert your database.

   Use the PROUTIL CONV1112 command, as shown:

```
proutil db-name -C conv1112
```

   CONV1112 performs extensive checks of your database during a pre-scan process. If the pre-scan encounters any issues that would prevent a successful conversion, the issues are logged and the convert utility exits without making any changes to the database (it is still a valid Release 11 database).

9. After you have successfully converted your database, back up your OpenEdge Release 12 database using Release 12 PROBKUP.

   - You should back up your Release 12 database in case it is damaged due to some failure. If you have only a Release 11 backup of your database, then you will need to go through the conversion process again.
   - Changes to large file support in OpenEdge Release 12.0 will impact databases created using CONV1112. See Database creation and large file support for more information.

Using the Schema Mover

While best practices advocate separating schema and user data, it is possible to have user data in the schema area. You can separate schema and user data by using PROUTIL dump and load or bulkload qualifiers, Database Administration, the Data Dictionary, or ABL code. However, as long as schema remains in an area, the extents continue to hold disk space, even after the removal of data. To free this disk space, use the Schema Mover qualifier to PROUTIL to move the schema. Once the area's schema is moved, user data may be recovered and the area then truncated.

To move schema to a new area:

1. Truncate the database's BI file. PROUTIL will send an error message if you do not.
2. Back up the database.

   Caution: PROUTIL with the MVSCH qualifier is a non-recoverable utility. If the execution fails, you cannot connect to the database and must restore a backup.

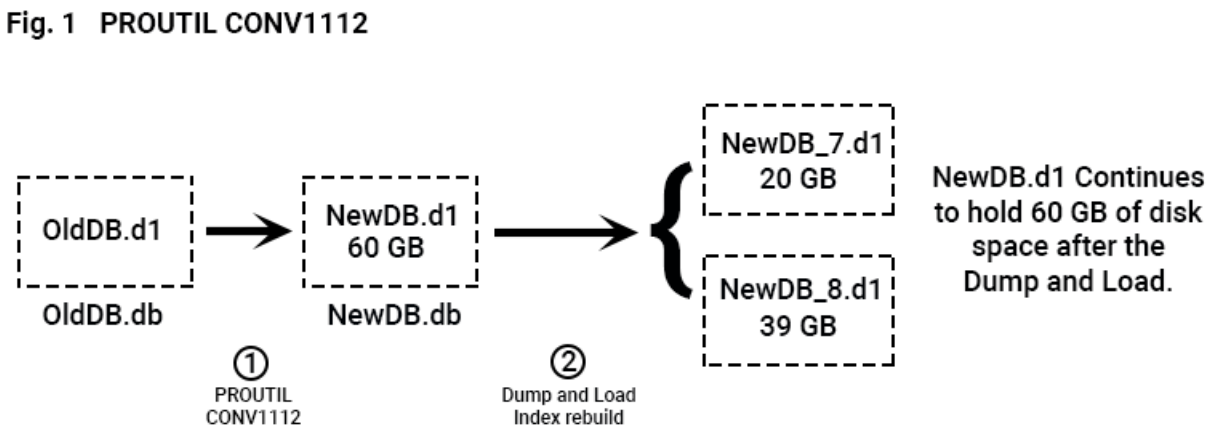
3. Execute the MVSCH command.

   Enter the MVSCH command as shown to begin the schema move, where *dbname* is the name of the converted database:

```
proutil dbname -C mvsch
```

As shown in Figure 1, after you convert your database to OpenEdge Release 12 using the PROUTIL CONV1112 command, Area 6 (`NewDB.d1`) contains both the database's schema and user data from the Release 11 database (`OldDB.d1`). Data may then be moved using PROUTIL binary dump and load, PROUTIL TABLEMOVE, or PROUTIL IDXMOVE. However, while schema remains in Area 6 (`NewDB.d1`), the area extents will continue to hold disk space, even after the data is moved.

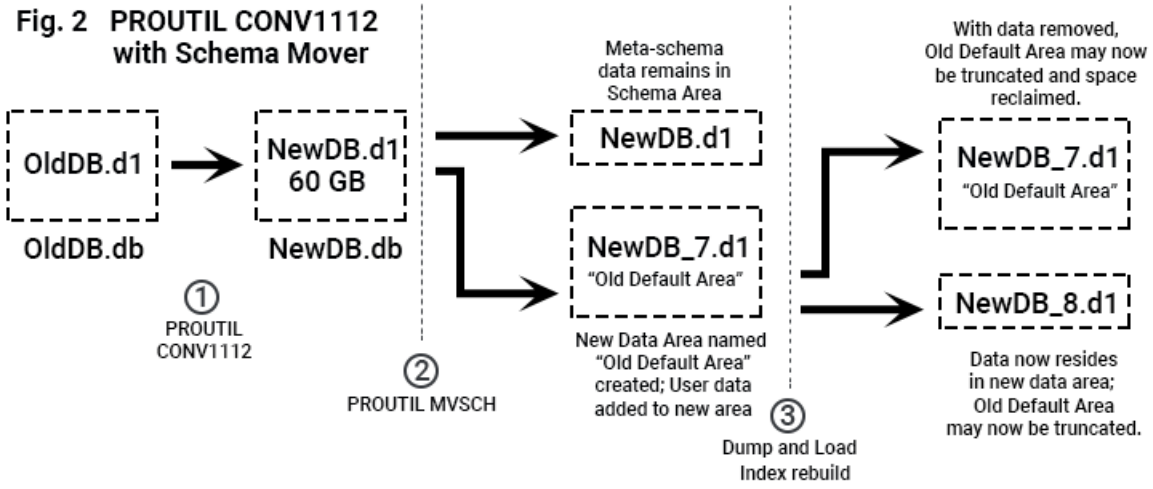
Figure 1: PROUTIL CONV1112 without Schema Mover.



When you initiate the schema move by entering the syntax shown in Step 3 above, PROUTIL recreates the schema area and moves meta-schema into the area, named `NewDB.d1`. PROUTIL then renames the old schema area `NewDB_7.d1` as "Old Default Area." User data from the old schema are remains in this area. With schema removed from the "Old Default Area," remaining user data may now be recovered and moved, and the area truncated to recover any unused space.

Note: Schema move only removes the schema from the "Old Default Area." It does not remove user data. User data must be moved before the area is truncated. Failure to move data prior to truncating the area will result in lost data.

Figure 2: PROUTIL CONV1112 with Schema Mover



Converting an OpenEdge Release 10 database to OpenEdge Release 12

To aid you in upgrading over multiple releases, OpenEdge provides you with utilities and a script to ease the database conversion process. The utilities are useful if you uninstall your older release of OpenEdge before completing the conversion, or if you are skipping over a release you never installed. The script assists you with the multi-step process of converting your Release 10 (10.1B and higher) database to Release 12.0.

- In your *install-dir*/bin directory, there are several directories with the name *dbutils. These directories contain the files needed to truncate BI files and run the conversion utilities, in the release that matches the directory name.

  Note: The permissions on the files in the *dbutils directories do not provide access to the "other" category or to non-admin users on Windows. Verify the permissions and adjust if necessary. The permissions must be defined in line with other permissions defined for users that administer the database. For example, if a database administrator group exists on UNIX, you must modify the group id for the files to that security group. On Windows, you must create a database administrator group and assign the permissions required to run *dbutils to the group.

- The *install-dir*/bin/102dbutils directory also contains a script, conv1012, to process the steps of truncating the BI files of a Release 10 database, converting the database to Release 11, and then converting the database to Release 12.

For more information, see the README file in the directory.

Database conversion utilities

OpenEdge Release 12 includes database utilities from previous releases to assist you in pre-conversion tasks if you have uninstalled your earlier installation. The following directories are included in your <*install-dir*>/bin directory:

- 83dbutils
- 91dbutils
- 101dbutils
- 102dbutils

The utilities in these directories allow you perform such tasks as truncating your BI file, converting a single-volume database to a multi-volume database, or performing a backup. See the readme file in each directory for details on the specific utilities available.

Copying a database

To copy a source database to a target database, use one of the following:

- PROCOPY utility
- PRODB utility
- An OpenEdge Data tool

These utilities copy the database structure as well as its contents. Consequently, a target database must contain the same physical structure as the source database. For example, it must have the same number of storage areas, records, blocks, blocks per cluster and block size.

Note: Do not use an operating system utility to copy an OpenEdge database.

See PRODB utility on page 841 for more information on using PRODB to copy a database. For more information about using the Data Dictionary or Data Administration tool in a graphical interface to copy a database, see the applicable online help system. For more information about using the Data Dictionary to copy a database in a character interface, see *OpenEdge Development: Basic Database Tools*.

Copying a database using PROCOPY

Use the PROCOPY utility to copy an existing database. For example, to copy the `Sports2000` database to a database named `mysports2000`, enter the following:

```
procopy Sports2000 mysports2000
```

PROCOPY supports storage areas. Therefore, if a target database exists, it must contain at a minimum the same type and number of storage areas and same extent types as the source database. However, the number of extents in the storage areas of the target database do not need to match the number of extents in the source database. PROCOPY attempts to extend the existing extents in the target database to accommodate the possible increase in size.

If a target database does not exist, PROCOPY creates one using an existing structure description (`.st`) file in the target database directory. If a `.st` file does not exist, PROCOPY creates the target database using the structure of the source database and places all of the extents in the same directory as the target database structure (`.db`) file, even when the source database resides in multiple directories.

Notes

- Changes to large file support in OpenEdge Release 12.0 will impact databases created using PROCOPY. See Database creation and large file support on page 42 for more information.

PROCOPY uses absolute pathnames

When you use the PROCOPY utility, the target database you create always has an absolute pathname regardless of the pathname convention used by the source database.

For example, if you use PROCOPY to create a database, name it `example1`, and use a relative path database such as `sports2000` as the source database, `example1` will have an absolute pathname even though the source database, `sports2000`, uses a relative pathname. Use PROSTRCT LIST to verify the absolute pathname of your target database, as shown:

```
procopy sports2000 example1
prostrct list example1
```

In the following sample output of the PROSTRCT LIST utility, note the absolute pathname of the database, `example1.db`:

```
Area Name: Control Area, Type 6, BlockSize 4096, Extents 1, Records/Block32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 32 KByte, Name: /usr1/V10/example1.db

Area Name: Primary Recovery Area, Type 3, BlockSize 8192, Extents 1
   Ext # 1, Type VARIABLE, Size 2176, Name: /usr1/V10/example1.b1

Area Name: Schema Area, Type 6, BlockSize 4096, Extents 1, Records/Block 32, Cluster
Size 1
   Ext # 1, Type VARIABLE, Size 960 KByte, Name: /usr1/V10/example1.d1

Area Name: Info, Type 6, BlockSize 4096, Extents 1, Records/Block 32, Cluster Size 1
   Ext #1, Type VARIABLE, Size 192 KByte, Name: /usr1/V10/example1_7.d1

Area Name: Order, Type 6, BlockSize 1024, Extents 2, Records/Block 32, Cluster Size 1
   Ext # 1, Type FIXED   , Size 1280, Name: /usr1/V10/example1_8.d1
   Ext # 2, Type VARIABLE, Size 0, Name: /usr1/V10/example1_8.d2
```

Deleting a database

Use the PRODEL utility to delete a database. For example:

```
prodel mysports2000
```

When you delete a database, PRODEL displays a message that it is deleting all files that start with *db-name* (the name of the database). PRODEL prompts you to confirm the deletions, depending on your system.

When you delete a database, PRODEL deletes all associated files and extents that were created using the structure description file (database, log, before-image and after-image files, and with two-phase commit, the transaction log file).

Note: The PRODEL utility does not delete the structure description file so that a file of your database structure is retained.

2

OpenEdge RDBMS Limits

This chapter lists the limits you need to know when configuring a database and supporting an application development environment.

For details, see the following topics:

- Database block sizes
- Maximum number of records per block
- Number and size of storage areas
- Table and index limits
- Number and size of sequences
- Maximum size of the primary recovery (BI) area
- Maximum database size
- Number of connections per database
- Number of simultaneous transactions per database
- OpenEdge database name limits
- File Handles
- Shared memory
- Data types and values

Database block sizes

The OpenEdge RDBMS supports the following database block sizes:

- 1024 bytes (1K)
- 2048 bytes (2K)
- 4096 bytes (4K)
- 8192 bytes (8K)

The default block size is 4K for Windows and LINUX, and 8K for UNIX.

Maximum number of records per block

You can define the maximum number of records per block for each application data area. When you define an area, you can specify 1, 2, 4, 8, 16, 32, 64, 128, or 256 records per block. If you do not explicitly specify the number of records per block when you define an application data area, the default number of records per block is:

- 64 if the block size is 8K
- 32 for all other block sizes

You cannot change the number of records per block for the schema data storage areas. It will always remain 32 for 1K, 2K, and 4K database block sizes and 64 for an 8K database block size. The records per block are only tunable for data areas.

Number and size of storage areas

An OpenEdge database supports a maximum of 32,000 storage areas, including 31,994 application data storage areas. Storage areas are identified by their names. The table below describes the area types.

Caution: The schema area can only be a Type I storage area, and is not recommended for data storage.

Table 3: Storage area types

Reserved area name	Contents	File extension
Control	Physical Database Structure	.db
Primary Recovery	Recovery Log Data	.b*n*
Transaction Log	Two-phase Commit Transaction Log	.t*n*
After Image	After-image Log Data	.a*n*
Schema	Schema Data	.d*n*
–	Application Data	.d*n*

The maximum size of a Type II storage area is fixed at approximately one petabyte **when large files are enabled**. A maximum of 1024 extents per area and a maximum size of one terabyte per extent, yield the maximum area size calculated as:

```
Maximum area size = maximum-number-of-extents-per-area * maximum-extent-size
                  =                1024                *       1TB
                  =                2^10                *       2^40
                  = 2^50 bytes
                  = 1PB
```

The maximum number of records per area is calculated with the following equation:

```
Maximum records per area = Maximum area size * records-per-block / block size
```

The table below lists the approximate theoretical maximum number of rows per Type II storage area, based on this calculation.

Note: The number of records per area is governed by the maximum area size rather than the addressable rows. Also take note of the reserved area recovery sizes detailed in Table 6: RDBMS reserved recovery space on page 65.

Table 4: Maximum rows per Type II storage area (approximate)

Database block size	Records per block	Records per area	Approximate maximum records per area (in M)[1]
8192 bytes(8K)	1	2^{37}	137,439 M
	2	2^{38}	274,878 M
	4	2^{39}	549,756 M
	8	2^{40}	1,099,512 M
	16	2^{41}	2,199,024 M
	32	2^{42}	4,398,048 M
	64 (default)	2^{43}	8,796,096 M
	128	2^{44}	17,592,192 M
	256	2^{45}	35,184,384 M

[1] **1 M = 1 million or 1,000,000**

Database block size	Records per block	Records per area	Approximate maximum records per area (in M)[1]
4096 bytes (4K)	1	2^{38}	274,878 M
	2	2^{39}	549,756 M
	4	2^{40}	1,099,512 M
	8	2^{41}	2,199,024 M
	16	2^{42}	4,398,048 M
	32 (default)	2^{43}	8,796,096 M
	64	2^{44}	17,592,192 M
	128	2^{45}	35,184,384 M
	256	2^{46}	70,368,768 M
2048 bytes (2K)	1	2^{39}	549,756 M
	2	2^{40}	1,099,512 M
	4	2^{41}	2,199,024 M
	8	2^{42}	4,398,048 M
	16	2^{43}	8,796,096 M
	32 (default)	2^{44}	17,592,192 M
	64	2^{45}	35,184,384 M
	128	2^{46}	70,368,768 M
	256	2^{47}	140,737,536 M

[1] **1 M = 1 million or 1,000,000**

Database block size	Records per block	Records per area	Approximate maximum records per area (in M)[1]
1024 bytes (1K)	1	2^{40}	1,099,512 M
	2	2^{41}	2,199,024 M
	4	2^{42}	4,398,048 M
	8	2^{43}	8,796,096 M
	16	2^{44}	17,592,192 M
	32 (default)	2^{45}	35,184,384 M
	64	2^{46}	70,368,768 M
	128	2^{47}	140,737,536 M
	256	2^{48}	281,475,072 M

The table below lists the maximum blocks per area and area size for Type I areas when large files are enabled.

Note: Unlike Type II storage areas that are governed by a fixed maximum area size, Type I storage area are governed by addressability of 32-bit database keys. Also take note of the reserved area recovery sizes detailed in Table 6: RDBMS reserved recovery space on page 65.

Table 5: Type I storage area maximums

Block size	Records per block	Maximum blocks per area	Maximum area size
8192 bytes(8K)	1	2,147,483,647	16TB
	2	1,073,741,823	8TB
	4	536,870,911	4TB
	8	268,435,455	2TB
	16	134,217,727	1TB
	32	67,108,863	512GB
	64 (default)	33,554,431	256GB
	128	16,777,215	128GB
	256	8,388,607	64GB

[1] **1 M = 1 million or 1,000,000**

Block size	Records per block	Maximum blocks per area	Maximum area size
4096 bytes(4K)	1	2,147,483,647	8TB
	2	1,073,741,823	4TB
	4	536,870,911	2TB
	8	268,435,455	1TB
	16	134,217,727	512GB
	32 (default)	67,108,863	256GB
	64	33,554,431	128GB
	128	16,777,215	64GB
	256	8,388,607	32GB
2048 bytes(2K)	1	2,147,483,647	4TB
	2	1,073,741,823	2TB
	4	53,687,0911	1TB
	8	268,435,455	512GB
	16	134,217,727	256GB
	32 (default)	67,108,863	128GB
	64	33,554,431	64GB
	128	16,777,215	32GB
	256	8,388,607	16GB

Block size	Records per block	Maximum blocks per area	Maximum area size
1024 bytes(1K)	1	2,147,483,647	2TB
	2	1,073,741,823	1TB
	4	536,870,911	512GB
	8	268,435,455	256GB
	16	134,217,727	128GB
	32 (default)	67,108,863	64GB
	64	33,554,431	32GB
	128	16,777,215	16GB
	256	8,388,607	8GB

Recovery sizes

The OpenEdge RDBMS reserves space within each area for recovery purposes. The reserved space allows room for database growth during recovery. A database area can grow up to its maximum size, less the reserve recovery space indicated in the table below.

Table 6: RDBMS reserved recovery space

Block size	Records per block	Reserved recovery space
1024	1,2,4,8,16,32,64	5G
	128,256	1G
2048	1,2,4,8,16,32,64	5G
	128,256	1G
4096	1,2,4,8,16,32,64,128,256	5G
8192	1,2,4,8,16,32,64,128,256	5G

As an area approaches its maximum size, less the reserve, warnings are issued at 80 and 90 percent of capacity.

Table and index limits

You cannot split a table or an index across storage areas, unless your database is enabled for multi-tenancy.

- In a database without multi-tenancy, each table and each index can be assigned to only one storage area.

- In a multi-tenant database, each table partition and each index partition can be assigned to one storage area. For more information on a multi-tenant database, see Multi-tenant database on page 347.

The size of a table or index (partition) is limited to the size of the storage area in which it resides.

The following limits exist:

- The maximum number of **table definitions** supported is 32,767, regardless of block size.
- Tables have a maximum number of **field definitions**: SQL supports 500, ABL supports 1000.
- The maximum number of **indexes** supported is 32,767, regardless of block size.
- Index entries have a maximum of 16 fields per index.
- Index size has the following constraints:
  - New databases with 4K and 8K block sizes, total variable-length storage requirements of all fields in an index entry must be less than 2000 characters.

    Note: Because the 2000 character limit includes storage overhead, the actual index key is limited to approximately 1970 characters.

  - Databases with 1K and 2K block sizes the maximum index entry size is approximately 200 characters.

Number and size of sequences

The number of unique sequences supported in a database varies by block size. An application that uses more than 250 sequences does not work on all database block sizes. The table below lists the number of sequences per database block size.

Table 7: Maximum number of sequences

Database block size	Maximum number of sequences
1024 bytes (1K)	250
2048 bytes (2K)	500
4096 bytes (4K)	1000
8192 bytes (8K)	2000

As of Release 12.0, all newly-created databases and databases converted by PROUTIL CONV1112 have 64-bit sequences and are bounded by the maximum of a signed 64-bit integer.

For more information on 64-bit sequences, see the online Help for the Data Dictionary or Data Admin.

Maximum size of the primary recovery (BI) area

Only the operating system or the size of your extents imposes a limitation on the size of a primary recovery (BI) area. The maximum possible size of a BI area is 32TB. The table below lists the maximum BI area size by block size used.

Table 8: Maximum primary recovery (BI) area size

Block size	Maximum BI area size
16,384 bytes (16K)	32TB
8192 bytes (8K) (default)	16TB
4096 bytes (4K)	8TB
2048 bytes (2K)	4TB
1024 bytes (1K)	2TB

Maximum database size

The maximum size of an OpenEdge database is determined by the:

- Number of storage areas
- Maximum size of a storage area

The maximum number of areas a database can support is 32,000. The first six areas are reserved, leaving 31,994 available data areas.

If fully utilized, the resulting database size is calculated as:

```
max database size = # of areas * maximum area size
                  =   31,994   *   1 petabyte
                  =  ~32,000 petabytes
```

Number of connections per database

The table below lists the maximum number of connections per database.

Table 9: Maximum number of connections per database

Database type	Limit
Single-user	1
Multi-user	Maximum number is machine dependent: up to 10,000 unless constrained by semaphore limits, process limits, or machine performance.

Number of simultaneous transactions per database

The table below lists the maximum number of simultaneous transactions per database.

Table 10: Maximum number of simultaneous transactions

Database type	Limit
Single-user	1
Multi-user	1 per user (maximum users = 10,000)

OpenEdge database name limits

The table below lists the database name limits for each operating system.

Table 11: Database name limits

Name type	Limit
Database names	One to 11 characters, excluding the pathname. You cannot use a file extension.
Pathnames	One to 255 characters, including the database name.

Database names can consist of any combination of English letters and numbers, beginning with A-Z or a-z. They cannot include ABL or SQL reserved words, any accented letters, or the following special characters:

```
\  "  `  *  ;  |  ?  [  ]  (  )  !  {  }  <  >  @  +  =  :  ~
```

File Handles

The OpenEdge RDBMS uses *file handles* (a UNIX term, roughly equivalent to the number of open files) when reading and writing to the database and related files. Most operating systems limit the number of file handles a user process can allocate at one time.

Use the following formula to determine the number of file handles used:

```
H = Static Handles + (# of .dn files) + (# of .bn files) + (# of .an files)
```

The elements of the above equation are defined as follows:

- **`H`** — The number of file handles the OpenEdge RDBMS uses.
- **`Static Handles`** — The number of handles allocated for any OpenEdge database. The number of static file handles that the process requires depends on whether you are running a client process or a server process:
  - **Client** — Requires ten file handles (`PROMSGS` + `LG` + `DB` + `LBI + SRT + RCD + STDIN STDOUT` + 2). The file handles used for the input and output devices (`STDIN` and `STDOUT`) are allocated by the operating system.
  - **Server** — Requires five handles (`PROMSGS` + `LG` + `DB` + 2).

- **# of `.dn` files** — The number of DB files defined for the database.
- **# of `.bn` files** — The number of BI files defined for the database.
- **# of `.an` files** — The number of AI files defined for the database.

Additional file handles are consumed as follows:

- If you are running a server in a UNIX environment that uses sockets for interprocess communication, add one file handle for each user.
- Application programs use additional file handles when reading and writing text files and when compiling programs. The maximum number of file handles supported by the AVM (ABL Virtual Machine) is 256.

Shared memory

The OpenEdge RDBMS uses shared memory to hold database buffers, latches, and control information including the lock table, AI buffers, and BI buffers. OpenEdge Release 10 can address larger amounts of allocated shared memory than previous releases. On 32-bit systems, the addressable limit is 4 gigabytes and on 64-bit systems, the addressable limit is 8 terabytes. These limits are architectural limits and are reduced in practice by limitations imposed by the underlying operating system and other factors. This is particularly true on 32-bit systems because the entire process address space is limited to 4 gigabytes and you cannot use all of it for shared memory. In practice, most 32-bit OpenEdge releases are limited to a maximum of approximately 2 gigabytes of shared memory.

The amount of shared memory allocated for database buffers is specified by the `-B` startup parameter. The theoretical maximum value for `-B` is 125,000,000 for 32-bit platforms and 1,000,000,000 for 64-bit platforms, but in practice, you will never successfully set `-B` that high.

The maximum number of shared memory segments is:

- 32 on 32-bit systems
- 256 on 64-bit systems

On UNIX, the number of shared memory segments, and their maximum size are constrained by kernel parameters. Fewer large segments tend to give better performance than many small segments.

Note: Actual maximum values are constrained by available system resources and operating system settings. For more information on shared memory segments, see Shared memory allocation on page 297.

Shared memory interactions

Due to the incompatibility of 64-bit and 32-bit shared memory versions, on platforms (operating system and architecture) where OpenEdge Release 11 supports both 64-bit and 32-bit versions, and both products can be simultaneously installed, note the following:

- Both the 32-bit and 64-bit version of OpenEdge on one platform, can access the same OpenEdge database. There is no need to convert or do a dump and load.
- Shared memory is not supported in a mixed bit mode. If a 64-bit server is started against the database, 32-bit clients cannot connect using shared memory. 32-bit clients can connect to the 64-bit server using client/server connections. The reverse is also true: if a 32-bit server is started against the database, 64-bit clients cannot connect using shared memory. The 64-bit clients have to connect using client/server connections.

- Mixing of servers is also not supported. The database can only have 32-bit servers or 64-bit servers running, never some of each.

Data types and values

The table below lists the OpenEdge SQL data types and value limits.

Table 12: SQL data type limits

SQL data type	Limit
BIGINT	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
BINARY	2000 bytes
BIT	0 or 1
BLOB	1GB
CHAR	2000 characters
CLOB	1GB
DATE	Year: 1 to 9999; Month: 1 through 12; Date: 1 through the last day of the month
DECIMAL	Defined in terms of precision and scale; precision=number of digits; scale=number of digits to the right of the decimal point **Note:** The scale cannot be greater than precision—precision is limited to 50; scale is limited to 10
DOUBLE PRECISION	2.2250738585072014E–308 through 1.7976931348623157E+308
FLOAT	2.2250738585072014E–308 through 1.7976931348623157E+308
INTEGER	–2,147,483,648 to 2,147,483,647
NUMERIC	Defined in terms of precision and scale; precision=number of digits; scale=number of digits to the right of the decimal point **Note:** The scale cannot be greater than precision—precision is limited to 50; scale is limited to 10
REAL	1.175494351E–38F to 3.402823466E+38F
SMALLINT	–32,768 to 32,767
TIME	00:00:00 to 23:59:59

SQL data type	Limit
TIMESTAMP	Combination of Date and Time limits
TINYINT	–128 to 127
VARBINARY	31,995 bytes.
VARCHAR	31,995

The table below lists ABL data types and value limits.

Table 13: ABL data type limits

ABL data type	Limit
BLOB	1GB
CHARACTER	Constrained by record size. **Note:** If a field has more than 32k bytes, you must write your own dump/reload procedure because the OpenEdge dump/reload procedure cannot handle fields larger than 32k bytes.
CLOB	1GB
DATE	1/1/32768 B.C. to 12/31/32767 A.D.
DATE-TIME	Same limit as DATE. TIME limit is: 00:00:00 to 23:59:59
DATE-TIME-TZ	Same limit as DATE-TIME. Time zone (TZ) limit is:–14:00 to +14:00
DECIMAL	50 digits total; 1 to 10 decimal places
INTEGER	–2,147,483,648 to 2,147,483,647
INT64	–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807
LOGICAL	TRUE/FALSE, YES/NO

Note: Data columns created using the OpenEdge SQL environment and having a data type that is not supported in an ABL environment are not accessible by ABL applications. Data columns created using an ABL environment can be accessed by OpenEdge SQL applications and utilities. Arrays of data can contain a maximum of 255 elements.

The table below describes data types supported by ABL and their corresponding SQL data types.

Table 14: ABL and SQL data type correspondence

ABL data type	OpenEdge SQL data type
CHARACTER	VARCHAR
DATE	DATE
DECIMAL	DECIMAL or NUMERIC
INTEGER	INTEGER
INT64	BIGINT
LOGICAL	BIT
RAW	VARBINARY
RECID	INTEGER
DATE-TIME	TIMESTAMP

3

Starting Up and Shutting Down

This chapter describes how to start up and shut down an OpenEdge database.

For details, see the following topics:

- Unified Broker Framework
- Starting a server or broker
- Threaded Database Server Functionality Delivered on Windows NT64, AIX64, and Linux64 Platforms
- Starting and stopping background writers
- Stopping a server or broker

Unified Broker Framework

The Unified Broker Framework provides a consistent interface for managing all OpenEdge RDBMS products installed on your network, and consists of the following elements:

- **AdminServer**

  An AdminServer provides secure administrative access to OpenEdge server products. An AdminServer is installed on every system where you install an OpenEdge database. The AdminServer grants access to each instance of an installed OpenEdge product. The AdminServer must be running in order to use OpenEdge Explorer, OpenEdge Management, or the command-line configuration utilities to manage your database.

  In Windows-based systems, the AdminServer starts automatically and runs as a service. For UNIX-based systems, a command-line utility (PROADSV) is used to start and stop the AdminServer. For more information about the AdminServer, see *OpenEdge Getting Started: Installation and Configuration*.

- **Command-line configuration utilities**

  The command-line configuration utilities allow you to start, stop, and configure installed OpenEdge RDBMS components. The database related command-line configuration utilities include:

- **DBMAN** — Starts, stops, and queries the current configuration of an OpenEdge database.
- **PROADSV** — Starts-up and shuts-down an AdminServer on UNIX. For more information about the AdminServer, see *OpenEdge Getting Started: Installation and Configuration*.

OpenEdge provides several tool options for managing your database configuration, as discussed in the following sections:

- OpenEdge Management on page 74
- OpenEdge Explorer on page 74

OpenEdge Management

OpenEdge Management is a browser-based management tool that you can use to monitor databases, files, networks, OpenEdge components, and system resources in an OpenEdge environment. For instructions on configuring your database to be managed by OpenEdge Management, see *OpenEdge Management and OpenEdge Explorer: Configuration*, *OpenEdge Management: Database Management*, or the OpenEdge Management online Help.

OpenEdge Explorer

OpenEdge Explorer is a browser-based configuration tool that allows you to set configuration properties for various OpenEdge resources, as well as to start and stop them, view their status, and view their log file data. For more information, see *OpenEdge Management and OpenEdge Explorer: Configuration* or the OpenEdge Explorer online Help.

Managing database configurations

The database configurations you create and/or manage with OpenEdge Management, OpenEdge Explorer, or the command-line tools, are saved in the `conmgr.properties` file. It stores the database, configuration, and server group properties. When you use OpenEdge Management, OpenEdge Explorer, or the DBMAN utility to start a database, a separate process interprets the information in the properties file and starts and stops whatever server the configuration specifies.

The `conmgr.properties` file resides in the `properties` subdirectory of the OpenEdge install directory.

Caution: Do not edit the `conmgr.properties` file directly. Instead, use OpenEdge Management or OpenEdge Explorer to create and edit database configurations.

Using the DBMAN command-line utility

After you use OpenEdge Explorer or OpenEdge Management to create the database configuration and store it in the `conmgr.properties` file, use the DBMAN command-line utility to start, stop, or query a database, as shown:

```
dbman [-host host-name -port port-number | service-name -user user-name]
                -database db-name [-config config-name -start | -stop | -query]
```

The DBMAN command-line utility supports the following parameters:

`-database` *db-name*

Specifies the name of the database you want to start. It must match the name of a database in the `conmgr.properties` file.

`-config` *config-name*

Specifies the name of the configuration with which you want to start the database.

`-start`

Starts the database *db-name* as defined by the configuration *config-name*.

`-stop`

Stops the database *db-name*.

`-query`

Queries the Connection Manager for the status of the database *db-name*.

`-host` *host-name*

Identifies the host machine where the AdminServer is running. The default is the local host. If your AdminServer is running on a remote host, you must use the `-host`*host-name* parameter to identify the host where the remote AdminServer is running.

`-port` *port-number* | *service-name*

Identifies the port that the AdminServer is listening on. If your AdminServer is running on a remote host, you must use the `-port` *port-number* parameter to identify the port on which the remote AdminServer is listening. The default port number is 20931.

`-user` *user-name*

If your AdminServer is running on a remote host, you must use the `-user` *user-name* parameter to supply a valid user name for that host. You will be prompted for the password.

Starting a server or broker

The server process coordinates all the database requests from all the users using a single database. The main database server is called the *broker*. The broker process manages shared resources and starts servers for remote users, if necessary.

Use OpenEdge Explorer, OpenEdge Management, or the PROSERVE startup command to start the server process, as shown:

```
proserve -db db-name | -servergroup server-group-name [parameters]
```

`-db` *db-name*

Specifies the database you want to start a server for (`-db` is implicit).

`-servergroup` *server-group-name*

Specifies the logical collection of server processes to start. The *server-group-name* you specify must match the name of a server group in the `conmgr.properties` file. You create server groups using OpenEdge Explorer or OpenEdge Management, which are saved in the `conmgr.properties` file.

parameters

Specifies the startup parameters for the broker/server. See Database Startup Parameters on page 473 for a list of broker/server startup parameters.

For more information about the PROSERVE command see PROSERVE command on page 467.

Specifying international character sets

An international database has one character set (code page) associated with all its data. This *database character set* information is stored in the database.

In addition, a database server has an *operating character set*. The operating character set is used for every character operation such as compare and substring. You can use the Internal Code Page (`-cpinternal`) international startup parameter to define the operating character set. If you do not use `-cpinternal`, the default operating character set is iso8859-1.

Specifying the operating character set

Use OpenEdge Explorer or OpenEdge Management to look up the operating character set name in the database configuration properties and reset it appropriately. For details, see *OpenEdge Management and OpenEdge Explorer: Configuration*.

To specify the operating character set from the command line, use the PROSERVE administration utility. For example:

```
proserve db-name -cpinternal character-set-name
```

Specifying the character set name of the database log file

To specify the output character set for database log messages, use OpenEdge Explorer or OpenEdge Management to look up the log character set (`LogCharacterSet`) name in the database configuration properties and reset it appropriately.

Otherwise, use the Log Character Set (`-cplog`) international startup parameter with the PROSERVE administration utility. For example:

```
proserve db-name -cplog character-set-name
```

For more information on character sets and character conversion, see PROUTIL Utility on page 629.

Network addressing using -S and -H

In all network environments, you use the Service Name (`-S`) startup parameter to assign a name to an OpenEdge broker/server. You then address this broker/server from a remote client by using the same value for `-S` as a startup or database connection parameter. Depending on your network type, you might also have to specify additional addressing criteria for remote clients. In terms of OpenEdge addressing, the Transmission Control Protocol (TCP) uses host addressing.

TCP requires a remote client to explicitly address the database server machine (or host) on which the server runs. In a TCP network, you must use the Host Name (`-H`) startup parameter to specify the host address. The `-H` value is the name assigned to the database server machine in your TCP/IP hosts file.

Use Host Name (`-H`) to identify the host name. For example:

`-H host-name`

The TCP/IP host-name (address) of the database server machine.

`-H localhost`

A reserved word that specifies that the database server communicates only with clients on the database server machine. Not applicable for DataServers.

Starting multiple brokers using the same protocol

You can start multiple brokers that use the same protocol. The `-Mn` parameter and the Maximum Servers per Broker (`-Mpb`) parameter, determine the number of servers a broker can start. In addition, you can use OpenEdge Explorer or OpenEdge Management to manage and configure server groups.

Use the following commands to start two brokers that use TCP and start multiple servers each:

```
proserve db-name -S service-name -H host-name -Mn n -Mpb n
proserve db-name -S service-name -H host-name -Mpb n -m3
```

db-name

Specifies the database you want to start. If the database is not in the current directory, you must specify the full pathname of the database.

`-S service-name`

Specifies the database server or broker process service name. You must specify the service name in a TCP network.

`-H host-name`

Specifies the machine where the database server runs.

`-Mn n`

Specifies the maximum number of remote client servers and login brokers that the broker process can start.

`-Mpb n`

Specifies the number of servers that the login broker can start to serve remote users. This applies to the login broker that is being started.

`-m3`

Starts the secondary login broker.

For example, you would use the following commands to start two brokers that use TCP and start four servers each:

```
proserve db -S demosv1 -H myhost -Mn 9 -Mpb 4
proserve db -S demosv2 -H myhost -Mpb 4 -m3
```

As the example shows, the `-Mn` value must be large enough to account for each additional broker and all servers. If you do not specify `-Mpb`, the value of `-Mn` becomes the default.

You must include the `-m3` parameter with every secondary broker startup command. While the `-Mpb` parameter sets the number of servers a broker can start, the `-m3` parameter actually starts the secondary broker.

If you start multiple brokers, you should also run the Watchdog process (PROWDOG). PROWDOG enables you to restart a dead secondary broker without shutting down the database server. For more information on PROWDOG, see PROWDOG command on page 471.

Accessing a server behind a firewall

The OpenEdge RDBMS allows you to use the Minimum Dynamic Server Port (`-minport`) and the Maximum Dynamic Server Port (`-maxport`) server startup parameters to provide client access to a server that is behind a firewall. This communication is possible only when access to the server can be limited. You supply this limit when you specify a group of port numbers with the `-minport` and `-maxport` parameters.

For example, suppose you start the following two login brokers:

```
proserve db -S demosv1 -H myhost -minport 4000 -maxport 4040
proserve db -S demosv2 -H myhost -minport 4041 -maxport 4080 -m3
```

A client requesting a connection from the first broker, `demosv1`, is assigned a port number in the range of 4000 to 4040. The 4000-to-4040 range limits access to the server by limiting communication to just 41 ports.

The default for -minport is 1025 for all platforms. Ports lower than 1025 are usually reserved for system TCP and UDP. The default for -maxport is 2000 for all platforms. Remember that some operating systems choose transient client ports in the 32,768-to-65,535 range. Choosing a port in this range might produce unwanted results.

Starting database brokers for SSL connections

OpenEdge supports Secure Sockets Layer (SSL) connections to the database server. SSL connections provide authentication and data privacy between the database server and clients according to Public Key Infrastructure (PKI) standards.

For a database server to start up with SSL, it must have access to:

- The private key that corresponds to the digital certificate the server uses to assert its identity to an SSL client
- A valid digital certificate that asserts the server's identity and contains the Public Key corresponding to the private key

---

Note: SSL incurs heavy performance penalties, depending on the client, server, and network resources and load. For more information on SSL and the security features of OpenEdge, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

Use the following command to start a broker that spawns SSL servers and accepts connections from SSL clients:

```
proserve db-name -S service-name [-H host-name] -ssl
      [-keyalias key-alias-name]
      [-keyaliaspasswd password]
      [-nosessioncache] [-sessiontimeout n]
```

db-name

Specifies the database you want to start. If the database is not in the current directory, you must specify the full pathname of the database.

-S *service-name*

Specifies the broker process service name.

-H *host-name*

Specifies the machine where the database server runs.

-ssl

Specifies that all database and SQL client connections will use SSL.

-keyalias *key-alias-name*

Specifies the alias name of the SSL private key/digital certificate key-store entry to use. The default is default_server.

`-keyaliaspasswd` *password*

Specifies the SSL key alias password to use to access the server's private key/digital certificate key-store entry. The default is the encrypted value of `password`. If you use a value other than the default, it must be encrypted. You can use the `genpassword` utility, located in your installation's `bin` directory, to encrypt the password.

`-nosessioncache`

Specifies that SSL session caching is disabled. Session caching allows a client to reuse a previously established session if it reconnects before the session cache time-out expires. Session caching is enabled by default.

`-sessiontimeout` *n*

Specifies in seconds the length of time an SSL session will be held in the session cache. The default is 180 seconds.

Threaded Database Server Functionality Delivered on Windows NT64, AIX64, and Linux64 Platforms

Under the classic model (`_mprosrv`), remote clients are processed individually by the database server. Using the new threaded model (`_mtprosrv`), remote client requests are processed concurrently. A threaded server can spawn and manage client threads up to a maximum of the value specified by the `-Ma` parameter.

In Release 12 of OpenEdge, the threaded database server has become the default server when a database is started in client server mode.

Initiating a Threaded Server Process

Initiate a threaded server process using:

```
proserve dbname -S sname -H hname -threadedServer n -threadedServerStack n
```

The following table summarizes the parameters `-threadedServer` and `-threadedServerStack`:

Parameter	Data Type	Characteristics
`-threadedServer`	Integer	Initiates the threaded database server Maximum value: 1 - Enabled Minimum value: 0 - Disabled Default value: 1
`-threadedServerStack`	Integer	Defines the thread stack size for threaded database servers in Kb. Maximum value: 2147483648 Minimum value: 64 Default value: 512

When `-threadedServer 1` is specified, the broker will spawn a threaded database server. The executable name of the threaded database server is `_mtprosrv` as opposed to `_mprosrv` for the classic database server.

Monitoring Threaded Servers and Remote Connections

Threaded servers may be monitored using the Promon command and identified in the _Content VST table. Threaded servers are identified in the **Type** field of the Promon report and the _Connect VST table as **TSRV**. Remote client connections to the threaded server are displayed in the following fields:

Field name	Data type	Description
SPID	Integer	Identifies the Server Process I.D. of the remote client connection. The SPID matches the Process ID (PID) of the threaded server (TSRV).
STID	Integer	Identifies the Server Thread I.D. of the connected remote client. The STID is unique to each connected remote client.

Remote client connections may be terminated using the PROSHUT command. Released threads will be reassigned by the threaded server to new remote client connections.

Starting and stopping background writers

Background writers improve performance by continually performing overhead functions in the background. Background writers are only available for Enterprise systems. You must manually start these background writers.

There are three types of background writers: asynchronous page writers (APWs), before-image writers (BIWs), and after-image writers (AIWs). The following sections explain how to start up and shut down background writers. See Managing Performance on page 275 for detailed information about background writers.

Starting and stopping an APW

A database can have between one and nine APWs running simultaneously. The optimal number is highly dependent on your application and environment. Start with two APWs and monitor their performance with PROMON. If there are buffers being flushed at checkpoints, add an additional APW and recheck. Applications that perform fewer changes to a database require fewer APWs. For more information on APWs and how to use PROMON to monitor them, see Using APWs to improve performance on page 283.

Note: If you perform no updates, no page writers are required.

To start an APW process, use OpenEdge Explorer, OpenEdge Management, or enter the following command on the local machine:

```
proapw db-name
```

Each APW counts as a process connected to a database and uses resources associated with a user. You might have to increase the value of the Number of Users (-n) parameter to allow for APWs. However, APWs are not counted as licensed users.

Stop an APW by disconnecting the process with the PROSHUT command.

For detailed information on the PROAPW and PROSHUT commands, see Startup and Shutdown Commands on page 461.

Starting and stopping a BIW

You can only run one BIW per database. You can start and stop the BIW process at any time without shutting down the database.

To start the BIW process, use OpenEdge Explorer, OpenEdge Management, or enter the following command on the local machine:

```
probiw db-name
```

The BIW counts as a process connected to a database and uses resources associated with a user. You might have to increase the value of the Number of Users (-n) parameter to allow for the BIW. However, the BIW is not counted as a licensed user.

Stop the BIW by disconnecting the process with the PROSHUT command.

Starting and stopping an AIW

You can run only one AIW process per database at a time. You can start and stop an AIW at any time without shutting down the database.

To start the AIW process, use OpenEdge Explorer, OpenEdge Management, or enter the following command:

```
proaiw db-name
```

You must have after-imaging enabled to use an AIW. For more information on after-imaging, see After-imaging on page 139.

The AIW counts as a process connected to a database and uses resources associated with a user. You might have to increase the value of the Number of Users (-n) parameter to allow for the AIW. However, the AIW is not counted as a licensed user.

To stop the AIW process, disconnect it with the PROSHUT command.

Stopping a server or broker

Before you turn off your computer or back up the database, you must shut down the server or broker process. Before you shut down the server, all application users must quit their sessions. If necessary, you can disconnect all users with the PROSHUT command's Disconnect a User or Unconditional Shutdown qualifiers. For more information, see PROSHUT command on page 83.

You can shut down a database using:

- OpenEdge Management or OpenEdge Explorer

- DBMAN utility
- PROSHUT command
- PROMON utility

To shut down the database, you must either be the user who started the database or have root privileges.

Note: Do not use operating system commands to shut down the database.

PROSHUT command

To shut down a database server with the PROSHUT command, enter one of the following:

```
proshut db-name [ -b | -by | -bn | -H host-name | -S service-name |
                  -shutdownTimeout [ immed | maximum | n [ h | m | s ] ]
                                   -F | -Gw
```

db-name

Specifies the database the server is running against.

-b

Indicates a batch shutdown will be performed. When no client is connected, the database automatically shuts down. When one or more clients are connected, PROSHUT prompts the user to enter "yes" to perform an unconditional batch shutdown and to disconnect all active users; or "no" to perform a batch shutdown only if there are no active users. The -b parameter combines the functionality of the -by or -bn parameters.

-by

Directs the broker to perform an unconditional batch shutdown and to disconnect all active users.

-bn

Directs the broker to perform a batch shutdown only if there are no active users.

-H *host-name*

Specifies the machine where the database server runs. You must specify the host name if you issue the shutdown command from a machine other than the host.

-S *service-name*

Specifies the database server or broker process service name. A TCP network requires the -S parameter.

-shutdownTimeout [immed | maximum | *n* [h | m | s]]

Specifies the amount of time for normal database activity to stop and users be disconnected before immediately shutting down (killing users still connected). If all normal activity ceases before the time

out value is reached, normal shutdown proceeds. Table 62 describes the shutdown time out parameter. If not specified, the default is 10 minutes. The minimum shutdown time out value is 60 seconds (`immed`) and the maximum time out value is 24 hours (`maximum`). The `-shutdownTimeout` parameter is ignored when forced shutdown (`-F`) is specified.

`-F`

Forces an emergency shutdown, on UNIX systems only. To use this parameter, you must run PROSHUT on the machine where the server resides. This parameter is not applicable for remote shutdowns or DataServer shutdowns.

Using `-by` with `-F` causes an emergency shutdown.

`-Gw`

For DataServers, specifies the DataServer broker to shut down.

For complete PROSHUT syntax, see PROSHUT command on page 468.

When you enter the PROSHUT command without the `-by`, `-bn`, or `-F` parameters, the PROSHUT menu appears:

```
1 Disconnect a User
2 Unconditional Shutdown
3 Emergency Shutdown (Kill All)
x Exit
```

The table below lists the PROSHUT menu options and their actions.

Table 15: PROSHUT menu options

Option	Action
1	Prompts you for the number of the user you want to disconnect.
2	Stops all users and shuts down the database. If you have multiple servers, PROSHUT stops them all. To stop a specific server process, use the appropriate operating system command.
3	Prompts you to confirm your choice. If you cancel the choice, you cancel the shutdown. If you confirm the choice, PROSHUT waits for five seconds before taking any action, then displays the following message: `Emergency shutdown initiated...` PROSHUT marks the database for abnormal shutdown and signals all processes to exit. After 10 more seconds, PROSHUT kills all remaining processes connected to the database, and deletes shared-memory segments and semaphores. The database is in a crashed state. The database engine performs normal crash recovery when you restart the database and backs out any active transactions.This option is available only if the database is on the same machine where you are logged in.
x	Cancels the shutdown without taking any action.

If you want to run the shutdown command noninteractively and avoid the PROSHUT menu, issue the PROSHUT command using either of the parameters described in the table below.

Table 16: PROSHUT Parameters

Parameter	Action
Kill Users (`-by`)	Unconditional batch shutdown; kills all active users
Proceed If No Users (`-bn`)	Batch shutdown only if there are no active users

When using the shutdown command from a machine other than the host, in a TCP/IP network, you must use the Host Name (`-H`) and Service Name (`-S`) parameters. The Host Name is the machine where the database server is running. The Service Name is the name of the database server or broker process, as defined in the `/etc/services` file on UNIX. For example, the following command shuts down the sports database from a remote machine in a BSD UNIX network:

```
proshut sports -H host-name -S sports-broker -by
```

Normal shutdown attempts to ensure that all database activity has ceased before initiating an immediate shutdown. The database is considered "active" if any of the following is true:

- There is forward movement of the before-image "cursor" indicating database update activity.
- There are active database connections.
- There are live transactions that have not been backed out.

Normal shutdown backs out live transactions and disconnects inactive connections.

Specifying `-shutdownTimeout` allows the database a specified amount of time to halt activity before initiating immediate shutdown. The database is considered "active" if any of the following is true:

- There is forward movement of the before-image "cursor" indicating update activity.
- There are active connections (excluding asynchronous page writers, the watchdog, the after-image management daemon, Replication, shutdown, and PROMON).
- There are live transactions that have not been backed out

Normal shutdown proceeds until all activity ceases or the specified amount of time elapses. At the end of the time out interval, the database is immediately shutdown by stopping all database output and killing all active users.

By default, normal shutdown has a maximum of 10 minutes to cease database activity and shutdown cleanly. The shutdown time out can range from one minute to 24 hours. For example, to give shutdown 1 hour to complete, you would specify the following:

```
proshut mydb -by -shutdownTimeout 1h
```

For more information on PROSHUT, see PROSHUT command on page 468.

PROMON shut down database option

You can use the PROMON utility to stop the database or disconnect any subset of users.

To shut down a database using PROMON:

1. Enter the following PROMON command: `promon` *db-name*

   When you start the monitor, the **Database Monitor** main menu appears:

```
   OpenEdge MONITOR Release 11
   Database: /usr/WRK_DBS/docsample
   1.  User Control
   2.  Locking and Waiting Statistics
   3.  Block Access
   4.  Record Locking Table
   5.  Activity
   6.  Shared Resources
   7.  Database Status
   8.  Shut Down Database
   9.  Currently Connected Tenants
 R&D.  Advanced options
   T.  2PC Transactions Control
   L.  Resolve 2PC Limbo Transactions
   C.  2PC Coordinator Information
   J.  Resolve JTA Transactions
   M.  Modify Defaults
   Q.  Quit
 Enter your selection:
```

2. Choose option 8, **Shut Down Database**.

   The shows an example of this option's output:

```
Enter your selection:  8
Usr    PID      Time of login     Userid  tty             Limbo?
1      6358     Dec 14 15:10:52   sue     /dev/ttyp0      no
4      7007     Dec 14 15:25:09   mary    /dev/ttyp5      no
                1 Disconnect a User
                2 Unconditional Shutdown
                3 Emergency Shutdown (Kill All)
                4 Exit
Enter choice>
```

3. Choose an option.

   If you choose **1 (Disconnect a User)**, the system prompts you for a user number. Choose **2 (Unconditional Shutdown)** to stop all users and shut down the database. If you have multiple remote-user servers, this stops all the servers.

II

Protecting Your Data

For details, see the following topics:

- Backup Strategies
- Backing Up a Database
- Recovering a Database
- After-imaging
- Maintaining Security
- Auditing
- Transparent Data Encryption
- Replicating Data
- Failover Clusters
- Distributed Transaction Processing

4

Backup Strategies

Backup and recovery strategies work together to restore a database that is lost due to a system failure. It is important to develop backup and recovery strategies that you can follow consistently and effectively. This chapter lists the steps needed to develop effective backup plans.

For details, see the following topics:

- Identifying files for backup
- Determining the type of backup
- Choosing backup media
- Creating a backup schedule

Identifying files for backup

To correctly back up an OpenEdge database, you must archive all of the files associated with the database. Files you must back up are:

- Database (`.db`, `.d`*`n`*) and before-image files (`.b`*`n`*)

  These files contain data and recent transaction information. You must back up these files as a unit; you need all these files to recover a consistent database.

- After-image files (`.a`*`n`*)

  If you have enabled after-imaging, the after-image (AI) files contain information required to reconstruct a database if a database disk is lost or damaged. You roll forward these files to reprocess all transactions that occurred since the last backup. Archive each AI file when it is full or when the database is not in use. You must use an operating system backup utility.

- Database log file (`.lg`)

  Event log (LG) files contain dates and times of important database events. They also contain messages and other historical information to help you diagnose complex problems by understanding the circumstances surrounding a failure. Back up these files regularly as part of the normal system backup procedures. You must use an operating system backup utility.

- Database key store (`.ks`)

  If you have enabled transparent data encryption, the key store stores the Database Master Key (DMK) externally from the database. You cannot open an encryption-enabled database without the key store. Back up the key store as part of normal system backup procedures. You must use and operating system backup utility.

- Transaction log files (`.t`*n*)

  If you use two-phase commit, the transaction log files contain information used to resolve in-doubt two-phase commit transactions. Back up these files regularly as part of the database backup procedures. You must use an operating system backup utility.

- Application files and program library files (`.pl`)

  While not strictly part of your database, including application files in backup strategy helps ensure overall system resiliency. Back up these files regularly as part of database backup procedures. You must use an operating system backup utility.

  If you store database triggers in a program library, you should maintain the library to make sure it matches the database schema. When you back up the database, PROBKUP backs up the database schema and any schema references to ABL (Advanced Business Language) triggers, but it does not back up ABL trigger code that is referenced in the schema.

Determining the type of backup

You can choose to perform either an OpenEdge backup (the PROBKUP utility) or an operating system backup, based on the needs of your business. Table contrasts OpenEdge and operating system backups.

Table 17: Backup options

OpenEdge	Operating system
Online or offline	Offline only
Full or incremental	Full only

To determine whether an online or offline backup should be used, consider:

- Is the database active 24 hours a day, 7 days a week? Is it possible to shut down the database for backing up?

  If the database must run 24 hours a day, 7 days a week, an online backup is necessary. If it does not, an offline backup can be used.

To determine whether a full or incremental backup is best, consider:

- Does the entire database fit on one volume of backup media? If not, will someone be present to change volumes during the backup?

If the database fits on one volume or someone is present to change volumes, a full backup can be performed. If not, consider using incremental backups.

Using PROBKUP instead of an operating system backup offers these advantages:

- PROBKUP automatically backs up all files required to recover a consistent database.
- PROBKUP allows both online and incremental backups, in addition to offline and full backups.
- PROBKUP allows users access to the database during an online backup.
- PROBKUP automatically marks the database as backed up.
- The Restore (PROREST) utility lets you easily verify backups.

If you choose not to use PROBKUP, you can use an operating system backup utility, but you cannot perform online or incremental backups. The exception to this rule is splitting a disk mirror during a quiet point to perform an operating system backup on the broken mirror. For information on using quiet points in a backup, see Using database quiet points on page 102. Be sure that the backup utility you choose backs up the entire set of files. Backing up a partial database provides an invalid result.

Note: Regardless of the backup method chosen, perform a complete database restore to test all backups and validate that the database is correct.

Full backups

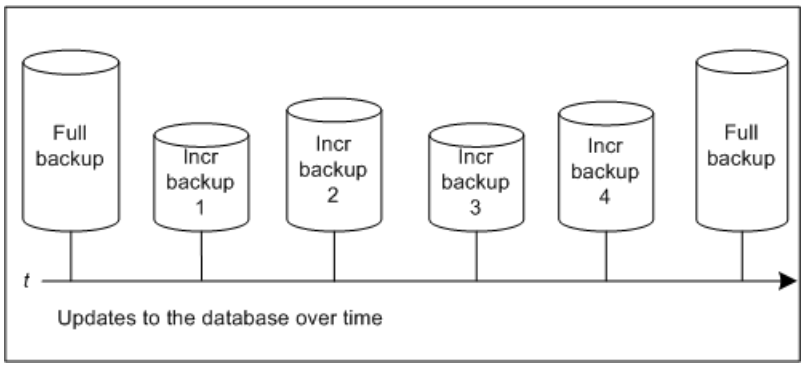
A full backup backs up all of the data of a database, including the BI files. You can perform a full backup using either the PROBKUP utility or an operating system utility. Ideally, you should perform a full backup of your database every day. However, depending on your recovery plan, you might decide to do less frequent full backups and more frequent incremental backups, or use after-imaging.

Incremental backups

An incremental backup backs up only the data that has changed since the last full or incremental backup. Incremental backups might take less time and media to back up the database; the amount of time you can save depends on the amount of data that has changed and the speed of your backup device. You must use PROBKUP to perform an incremental backup.

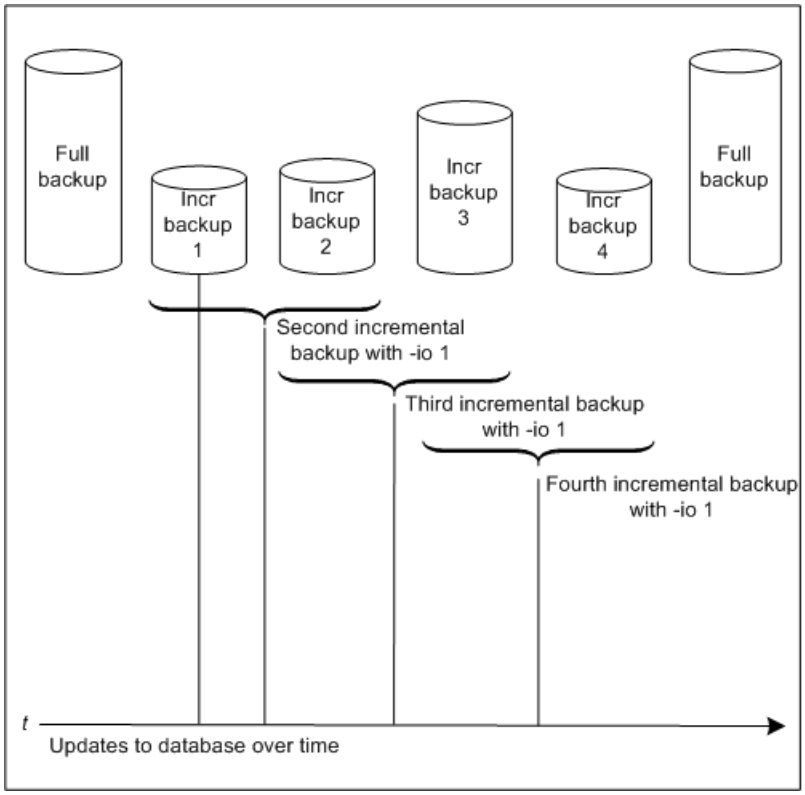
In an OpenEdge database, the master block and every database block contains a backup counter. The counter in the master block is incremented each time the database is backed up (online or offline, full or incremental). When a database block is modified, PROBKUP copies the backup counter in the master block to the backup counter in the modified database block. When you perform an incremental backup, PROBKUP backs up every database block where the counter is greater than or equal to the master block counter. The following figure illustrates how the size of an incremental backup varies, depending on the amount of changes to the database between incremental backups.

Figure 3: Incremental backup—size varies



The Overlap qualifier (`-io`) specifies the amount of redundancy in an incremental backup. When you specify Overlap (`-io`), PROBKUP backs up every database block where the counter is greater than or equal to the master block counter, less the overlap value. The following figure illustrates how the Overlap qualifier determines the data backed up during an incremental backup.

Figure 4: Incremental backup with overlap



For more information on the `-io` parameter, PROBKUP utility on page 832.

You must perform a full backup of a database before you can perform the first incremental backup. You should also perform full backups regularly in conjunction with incremental backups. If you do not perform full backups regularly, you will use increasing amounts of backup media for incremental backups and increase recovery time restoring multiple incremental backups in addition to the full backup.

Online backups

An online backup lets you back up the database while it is in use. You must use PROBKUP to perform online backups. Perform an online backup if the database cannot be taken offline long enough to perform an offline backup. You can perform both full and incremental online backups.

When deciding whether to use online backups, consider the following:

- You cannot perform an online backup on a system running in single-user mode.
- If you have enabled after-imaging, when you perform an online backup of a database, the database engine automatically switches over to the next AI file. Before you start an online backup, you must make sure that the next AI file is empty. If the file is not empty, PROBKUP aborts and notifies you that it cannot switch to the new file. For information about after-imaging, see After-imaging on page 139.
- If you want to enable after-imaging while your database is online, you must perform a full online backup. The resulting backup is the baseline for applying after-image extents. For more information about enabling after-imaging, see the Enabling after-imaging online on page 208.
- When you begin an online backup, database activity pauses until the backup header is written. Make sure that your backup media is properly prepared before issuing the PROBKUP command in order to minimize the duration of the pause. Until the database engine writes the header information, you cannot update the database. If you use more than one volume to back up the database, there is a similar delay each time you switch volumes.
- You cannot use the PROBKUP parameters Scan (`-scan`) or Estimate (`-estimate`) for online backups.

Offline backups

To perform offline backups, you must first shut down the database. If you perform an offline backup with an operating system utility on a database while it is running, the backup is invalid. You can use PROBKUP or an operating system utility to perform offline backups.

Choosing backup media

Depending on the operating system, there are many different options available for backup media. Available media includes:

- Disk files
- Tape cartridges
- Disk cartridges
- Writable disks (CDs and DVDs)

When choosing backup media, consider the media's speed, accessibility, location, and capacity for storage. The following table lists questions to ask yourself when choosing backup media.

Table 18: Backup media questions

To consider this ...	Ask this question ...
Storage capacity	Is the media large enough to contain all the files you need to back up?
Data transfer speed	Is the media fast enough to allow backup of files in a reasonable time?
Accessibility	Can you use the media at the time you need to run the backup?
Location	Is the device accessed through a network? If so, can the network handle the large volume of activity generated during a backup?

Creating a backup schedule

The backup schedule is a fundamental part of the recovery plan. It determines how frequently you must perform backups, assigns responsibility for making backups, and serves as a record of when backups are performed.

Ideally, you should perform a full backup of the database every day. However, depending on availability requirements and your recovery plan, a less frequent full backup and frequent incremental backups might be sufficient.

Several considerations impact the backup schedule:

- Database integrity on page 94
- Database size on page 94
- Time on page 95
- Unscheduled backups on page 95

Database integrity

To preserve database integrity, you must:

- Back up the database frequently
- Back up the AI files separately from the database backup
- Use the PROREST verify parameters Partial Verify (`-vp`) and Full Verify (`-vf`) to verify that a backup is valid

Database size

If the database is very large, it might be impractical to fully back up the database daily. You might choose to back up the database file every other day, or even once a week. Instead of frequent full backups, consider performing daily incremental backups or, if you have after-imaging enabled, only backing up the after-image files.

You can perform daily incremental backups. Incremental backups only back up the blocks that have changed since the previous backup. You can specify an overlap factor to build redundancy into each backup and help protect the database. However, you should also perform a full backup at least once a week to limit the amount of backup media used for incremental backups and to ease data recovery.

If you enable after-imaging, back up the after-image files every time you perform a backup. Immediately after performing a full or incremental backup, start a new after-image file. When you back up AI files, you back up whole transactions, but incremental backups back up just the blocks that have changed. As a result, AI file backups can use more space than incremental backups.

If you make many database updates and you are on a weekly full backup schedule, it is possible that the after-image files will grow very large during the week. If so, backup and empty the AI files every day. This daily backup approach keeps the AI file relatively small and ensures that the AI file is archived on a regular schedule. You can also use the After Image File Management Utility to archive and empty your AI files. For more information, see AI File Management utility on page 150.

Note: PROBKUP does not back up AI files. You must use an operating system backup utility.

Time

When creating a backup schedule, consider both the time required to perform backups and the time required to recover a database from the backups:

- Performing daily full backups of your system might require too much time. If you make few updates to the database each day, a daily full backup might be unnecessary. Thus, you might want to perform daily incremental backups and a weekly full backup. If you have after-imaging enabled, remember to back up the after-image files for both incremental and full backups.
- If you perform full backups less than once a week, you must maintain multiple incremental backups, which makes recovery more complicated and prone to operator error. Restoring a backup becomes a multi-step process of first restoring the last full backup, then restoring the subsequent incremental backups.
- If you enable after-imaging, you can perform daily backups of the after-image file instead of performing incremental backups. However, recovering from the AI file backups requires restoring the AI files then rolling forward through multiple after-image files. This is more time intensive than restoring a full backup with PROREST. Because backing up AI files backs up whole transactions instead of just the blocks that have changed since the most recent backup, restoring a database from incremental backups is quicker than restoring AI files and rolling forward the AI files.

Unscheduled backups

In addition to scheduled backups, you might have to perform additional backups for the following reasons:

- To run a large job with the No Crash Protection (`-i`) startup parameter—before running the job, back up the database and after-image files
- As part of the recovery process, to re-establish a valid environment
- Before and after any major changes to an application or database
- When installing a new release of the OpenEdge RDBMS

5

Backing Up a Database

Backing up your database is an important part of database maintenance. Regular backups provide a starting point for recovery of a database lost to hardware or software failure.

For details, see the following topics:

- Using PROBKUP
- Performing an offline backup
- Performing an online backup
- Using database quiet points
- Performing an operating system backup
- Database backup examples
- Verifying a backup
- CRC codes and redundancy in backup recovery
- Restoring a database

Using PROBKUP

Using the OpenEdge Backup utility (PROBKUP) you can perform an online full backup, an online incremental backup, an offline full backup, or an offline incremental backup. Which you use is determined by your backup plan. You can also enable after-imaging and AI File Management as part of a full online backup. The syntax below details the parameters to use with PROBKUP:

```
probkup [ online ] db-name [ incremental ] device-name
          [ enableai ] [ enableaiarchiver -aiarcdir dirlist
            [-aiarcinterval n] [-aiarcdircreate] ]
          [ bibackup { active | all } ]
          [   -estimate | -vs n | -bf n | -verbose | -scan
            | -io i | -com | -red i | -norecover
          ]
```

For more information on PROBKUP, see PROBKUP utility on page 832. For information on using PROBKUP to enable after-imaging and AI File Management, see After-imaging on page 139.

Performing an online full backup with PROBKUP

To perform an online, full backup, run PROBKUP from the operating system prompt using the following command syntax:

```
probkup online db-namedevice-name [parameters]
```

UNIX full backup example

The database administrator of Company X's development department performs a full backup of the `devel.db` database every Friday on 9-track tapes. The DBA prepares the backup media according to the operating system documentation, then performs a full online backup of the `devel.db`.

To perform a full online backup of devel.db:

1. Verify that the database is not in use.

   Enter the following command:

   ```
   proutil devel -C BUSY
   ```

2. Perform a full online database backup.

   Enter the following command:

   ```
   probkup online devel /dev/rrm/0m -vs 35 -bf 20 -verbose
   ```

The command, `devel` identifies the name of the database you are backing up; `online` specifies that the backup is an online backup; `/dev/rrm/0m` specifies the output destination is a tape drive; `-vs 35` indicates that the volume size in database blocks is 35; `-bf 20` specifies that the blocking factor is 20; and `-verbose` displays information at 10-second intervals during the backup. If you do not specify the volume size, PROBKUP fills the entire tape before prompting you for a new tape.

As the full offline backup of `devel.db` begins, the following report appears:

```
64 bi blocks will be dumped.
336 out of 336 blocks in devel will be dumped.
This will require 369664 bytes of backup media.
This backup will require a minimum of 400 blocks to restore.
1 volume will be required.
Backed up 400 blocks in 00:00:05.
Wrote a total of 18 backup blocks using 369664
bytes of media.
Backup complete.
```

The number of backup blocks is the number of `-bf` units written to the tape. Backup blocks contain data, primary recovery, and error-correction blocks.

This example backs up a very small database. Using the `-red` parameter on a larger database increases the amount of time and backup media required for the backup. Also, PROBKUP displays the number of blocks and the amount of backup required for an uncompressed database because you cannot specify the `-scan` parameter for an online backup.

3. If you enable after-imaging, back up the AI files to a separate tape or disk using a UNIX backup utility.

Note: If you enable after-imaging, OpenEdge automatically switches AI extents before beginning an online backup.

Backup Counter Increment limits using the PROBKUP Utility with Type I storage areas

The number of times a database featuring Type I storage areas may backed up using the PROBKUP utility is limited for databases operating on OpenEdge Release 11.7.5 or earlier. This limit is tracked by the Backup Counter Increment (bk_incr) reported in the Master Block. Exceeding this limit will result in a Backup Counter overflow and a corrupted database backup file.

The Backup Counter Increment limit is 65,535. In applicable terms, this limit amounts to approximately 22 months for a database that is backed up every 15 minutes. For a database that is backed up every hour, this limit would be reached in approximately 7.5 years.

To reset the Backup Counter Increment, a full dump and load of the database must be performed.

The limit for the Backup Counter Increment is higher in Type II storage areas. To avoid a Backup Counter overflow, Progress recommends moving all user data from the Schema area and Type I areas into Type II areas.

As of OpenEdge Release 12, the Backup Counter Increment has been increased, and users are unlikely to experience overflow. However, users performing a full online backup of databases featuring Type I storage areas may experience increased wait times. Moving data from Type I areas into Type II areas will eliminate any extended wait times. Performing more frequent incremental backups will also decrease these wait times.

Testing backups

Test backups regularly to ensure they are valid and available when you need them. Be sure to test the backup before you need it to restore a database. Ideally, you should test the backup immediately after you perform it. Then, if there are problems with the backup, you can make another backup copy. If you wait until you need the backup to test it, the database will no longer be available.

The following table lists the PROREST utility parameters you can use to test backups performed with PROBKUP.

Table 19: PROREST utility verification parameters

Parameter	Description
`-vp`	Specifies that the restore utility read the backup volumes and compute and compare the backup block cyclic redundancy checks (CRCs) with those in the block headers. To recover any data from a bad block, you must have specified a redundancy factor when you performed the database backup. See CRC codes and redundancy in backup recovery on page 111 for more information about error correction blocks and data recovery.You can use the Partial Verify parameter with both online and offline backups.
`-vf`	Specifies that the PROREST utility will compare the backup block-for-block to the database.

The Partial Verify and Full Verify parameters do not restore or alter the database.

For more information, see PROREST utility on page 844.

If you use an operating system utility to back up the database, with each backup, verify that you have backed up the entire database. The PROBKUP utility automatically backs up the appropriate files; with an operating system utility, you must make sure the files are included in the backup.

Archiving backups

Properly archiving backups helps you ensure database integrity. Follow these guidelines when archiving backups:

- Clearly label each backup volume. Information on the label should include:
  - The type of backup (incremental or full, online, or offline)
  - The date and time of the backup
  - The full pathname of the database
  - The volume number and total number of volumes of the media (volume 1 of 4, for example)
  - The initials of the person who performed the backup
  - The exact command used to back up the database
- Keep a minimum of 10 generations of full backups. Keep daily backups for at least two weeks, weekly backups for at least two months, and monthly backups for a year. Buying extra tapes is much less expensive than manually reconstructing lost databases.
- Keep backups in an area other than where the computer is located, preferably in another building. In the event of building damage, you are less likely to lose both the online and backup versions of the database.

Performing an offline backup

You can perform both full and incremental backups offline. You must perform a full backup before performing an incremental backup. Following the first full backup, you can perform any number of incremental backups. However, you should perform a full backup at least once a week.

To perform either a full or incremental offline backup:

1. Verify that the database is not in use.

   If you are performing an offline backup, the database must not be in use. The server must be shut down and no single-user session can be active. PROBKUP does not allow access to the database during a full offline backup.

   On UNIX systems that support file locking and where backup utilities (for example, `cpio`) honor file locks, an attempt to back up a database in use causes the utility to hang, waiting for the OpenEdge session to end. On all other systems, there is no way to prevent you from backing up a database in use.

   Use the following PROUTIL BUSY utility to determine if the database is in use:

   ```
   proutil db-name -C busy
   ```

   The BUSY qualifier returns a code indicating whether the database is in use. You can use the codes returned by the BUSY qualifier in scripts, files, or procedures. For more information see PROUTIL BUSY qualifier on page 648.

2. Shut down the database server.

   Before you back up the database, you must shut down the database server. See Starting Up and Shutting Down on page 73 for information about shutting down the database server.

3. Perform the offline backup.

   Use PROBKUP to perform either a full or incremental offline backup. Run PROBKUP from the operating system prompt, as shown:

   ```
   probkup db-name device-name [parameters]
   ```

   By default, PROBKUP performs a full backup. To perform an incremental backup, specify the incremental qualifier.

   As you begin the full backup of a database, a report appears on your terminal that indicates:

   - How many bytes are required on the backup media
   - How many active data blocks are written to the backup media
   - How many blocks are dumped
   - How many blocks are required to restore the database

   When the backup successfully completes, the report displays the total number of bytes on the backup media and how long it took to complete the backup.

   Note: If a system failure occurs while you are performing the full backup, perform the backup again.

Performing an online backup

Use PROBKUP to perform either a full or incremental online backup. Run PROBKUP from the operating system prompt. Use the following command:

```
probkup online db-name device-name [parameters]
```

By default, PROBKUP performs a full backup. To perform an incremental backup, specify the incremental qualifier:

```
probkup online db-name incremental device-name [parameters]
```

Note: Databases started with the No Crash Protection (`-i`) parameter cannot be backed up online. For databases with after-imaging enabled, online backup will automatically perform an AI extent switch. For more information on after-imaging, see After-imaging on page 139.

Improving performance of an online backup

The `bibackup` parameter to online backup optimizes the backup of BI clusters. By default, and if you specify `bibackup active`, PROBKUP starts by backing up the active BI cluster, and proceeds to the previous BI cluster in the BI cluster ring. PROBKUP with `bibackup active`, proceeds to each previous BI cluster, backing up a minimum of 4 BI clusters, and continuing to backup the BI clusters until a cluster with no active transactions is reached.

When evaluating performance considerations, the best case scenario is that the minimum of 4 BI clusters are backed up, and the worst is that all BI clusters are backed up. For more information on BI clusters, see Before-image I/O on page 288.

The `bibackup active` option is the default behavior. If you want to backup all BI clusters, specify `all` for `bibackup`, as shown:

```
probkup online db-name incremental device-name bibackup all
```

Note: PROREST restores the backup containing only active BI clusters without any changes to the command line. Online backup for Release 11.3, is not backward compatible unless you specify `bibackup all`.

Using database quiet points

If you use your operating system's disk mirroring to provide data redundancy as part of your backup and recovery strategy, you can use database quiet points, on systems with an Enterprise database license, to maintain database consistency during an OS mirror fracture or split operation on an active online database.

To maintain database consistency during an OS mirror fracture or split operation on an active online database:

1. Use the PROQUIET command to enable a database quiet point.

Enter the command with the syntax shown:

```
proquiet dbname enable
```

In this command, *dbname* specifies the name of the database for which you are enabling a database quiet processing point.

During a database quiet processing point all file write activity to the database is stopped. Any processes that attempt to start a transaction while the quiet point is enabled must wait until you disable the database quiet processing point.

2. Use an operating system utility to perform the OS mirror fracture or split operation.

   Upon successful completion of this command, the fractured disk contains a duplicate of the active online database.

3. Use the PROQUIET command to disable the database quiet point.

   Enter the command with the syntax shown:

   ```
   proquiet dbname disable
   ```

   In this command, *dbname* specifies the name of the database for which you are disabling a database quiet processing point.

   For more information and the complete syntax for PROQUIET, see Startup and Shutdown Commands on page 461.

4. Update the structure description (`.st`) file of the fractured version of the database. Replace the logical location reference (which still references the active database) with the physical location reference of the fractured mirror.
5. Run the PROSTRCT REPAIR command on the fractured version of the database with the updated `.st` file.

   Enter the command with the syntax shown:

   ```
   prostrct repair dbname [description-file]
   ```

   In this command:

   - *dbname* specifies the name of the database for which you are repairing the extent list and master block.
   - *description-file* specifies the name of the structure description (`.st`) file.

   Running PROSTRCT REPAIR updates the shared memory and semaphore identification information to reflect the offline status of the fractured version of the database, and update the file list information with the information in the updated `.st` file

6. Use the PROBKUP utility with the `-norecover` startup parameter to back up the fractured version of the database.

Enter the command with the syntax shown:

```
probkup dbname -norecover
```

In this command, *dbname* specifies the name of the of the fractured version of the database.

Note: The `-norecover` parameter prevents PROBKUP from performing crash recovery or switching to a new AI extent as part of the backup process. Use of the `-norecover` parameter is noted as an entry in the `.lg` file.

Performing an operating system backup

When performing a backup using an operating system utility instead of PROBKUP, you must perform the following extra steps:

- Be sure to back up all the proper files.
- Make sure that the database is not used during the backup. Otherwise, the backup will be invalid. If you have an Enterprise database license, you can do this by using the PROQUIET command to create a database quiet point. Regardless of your database license you can shut down the server and make any single-user session inactive.
- After you perform and verify the backup, mark the database as backed up.

Use the following steps to perform a backup using an operating system utility:

1. Shut down the database server.

   Before you back up the database, you must shut down the database server. See Starting Up and Shutting Down on page 73 for information about shutting down the database.

2. Verify that the database is not in use with the PROUTIL BUSY utility.

   The following example shows the syntax for PROUTIL BUSY:

   ```
   proutil dbname -C busy
   ```

   The BUSY qualifier returns a code indicating whether the database is in use. You can use the codes returned by the BUSY qualifier in scripts, files, or procedures. For detailed information, see PROUTIL BUSY qualifier on page 648.

3. Make a note of the last entry in the log file. You will use this information later to verify that the database is not used during the backup.
4. Back up the database.

   Use an operating system backup utility to back up the database files. Ensure that your backup technique backs up the entire file. On many UNIX systems, certain utilities (for example, `cpio`) back up only the first part of files that are larger than a specified size (controlled by the `ULIMIT` parameter). Backups of only the first portion of a database file are of no value.

5. Verify that the backup is valid.

First, compare the last entry in the log file against the entry you noted in Step 3. If an entry has been added to the log file since you checked in Step 3, the database might have been used. If the database was used during the backup, then the backup is invalid. You must perform another backup.

Second, verify that you have backed up the entire database. The PROBKUP utility automatically backs up the proper files; with an operating system utility, you must make sure the proper files are included in the backup. See Identifying files for backup on page 89 for the complete list of files.

6. Mark the database as backed up.

   After you have verified the backup, use the RFUTIL MARK BACKEDUP utility to mark the database as backed up, as shown:

```
rfutil dbname -C mark backedup
```

   For more information, see RFUTIL Utility on page 797.

Database backup examples

This section includes examples of how to use PROBKUP to perform database backups on UNIX and Windows.

Incremental backup example

This example shows how to use all of the possible incremental backup parameters to perform an incremental offline or online backup of the `devel.db` database. To perform an online backup, skip Step 2 through Step 4.

Use the following steps to perform an incremental offline backup of the devel.db database:

1. Prepare the backup media according to the operating system documentation.
2. Verify that the database is not in use with the PROUTIL BUSY COMMAND.

   Execute PROUTIL BUSY by entering the following command:

```
 proutil devel -C BUSY
```

3. Shut down the `devel.db` database.

   Enter the PROSHUT command:

```
proshut devel
```

4. Run PROBKUP `-estimate` to determine how much media is necessary for the backup.

   PROBKUP with the estimate command is as follows:

```
probkup devel incremental /dev/null -estimate
```

5. Perform the incremental backup.

   Enter the following command to perform an incremental offline database backup:

```
probkup devel incremental /dev/rrm/0m -vs 4000 -bf 20
    -verbose -io 1 -com -red 5 -scan
```

   Enter the following command to perform an incremental online database backup:

```
probkup online devel incremental /dev/rrm/0m -vs 35
    -bf 20 -verbose -io 1 -com -red 5
```

   The following parameters are utilized in the preceding commands:

   `devel`

   Identifies the name of the database you are backing up.

   `online`

   Specifies that the backup is an online backup.

   `incremental`

   Specifies that the backup is an incremental backup.

   `/dev/rrm/0m`

   Specifies the output destination is a tape drive, `/dev/rrm/0m`. If backing up to tape, do not use the same reel of tape that was used for the full backup.

   `-vs 35`

   Indicates that the volume size in database blocks is 35. If you do not specify the volume size, PROBKUP fills the entire tape before prompting you for a new tape.

   `-bf 20`

   Specifies that the blocking factor is 20.

   `-verbose`

   Displays information at 10-second intervals during the backup.

   `-io 1`

   Specifies that you can lose one incremental backup and still be able to restore the database. Specifies that all blocks that have changed since the backup before the last backup should be archived.

   `-com`

   Indicates that the data should be compressed before it is written to the tape drive. If you specify the `-com` parameter and do not use `-scan`, PROBKUP displays the number of blocks and the amount of backup media required for an uncompressed database.

`-red 5`

Specifies that PROBKUP creates one error-correction block for every five blocks that are backed up.

`-scan`

Allows the backup utility to scan the database before backing it up to determine the number of blocks to be backed up.

As the incremental offline backup of `devel.db` runs, the following report appears:

```
64 bi blocks will be dumped.
13 out of 336 blocks in devel will be dumped.
This will require 103424 bytes of backup media.
This backup will require a minimum of 400 blocks to restore.
1 volume will be required.
Backed up 77 blocks in 00:00:01.
Wrote a total of 5 backup blocks using 103424 bytes of media.
Backup complete.
```

The number of backup blocks is the number of `-bf` units written to the tape. Backup blocks contain data, BI, and error-correction blocks.

This example backs up a very small database. Using the `-red` parameter on a larger database increases the amount of time and backup media required for the backup.

As the incremental online backup of `devel.db` runs, the following report appears:

```
Incremental backup started.
Backed up 70 blocks in 00:00:01.
Wrote a total of 3 backup blocks using 103424 bytes of media.
Backup complete.
```

6. If you have after-imaging enabled, back up the AI files to a separate tape or disk using a backup utility.

Full backup example

The database administrator of Company X's development department performs a full backup of the `devel.db` every Friday, and incremental backups on other weekdays. Both full and incremental backups are done on disk and on a single disk file. The database is on a Windows system.

When choosing the parameters, consider that the backup takes significantly longer when you use the `-com` or `-red` parameters. If you use the `-red` parameter, the backup also uses more backup media.

The following procedure outlines how to perform an full online or offline backup of the `devel.db` database. To perform an online backup, skip Step 2 through Step 4 of the procedure.

Use the procedure that follows to perform a full backup of the devel.db database:

1. Prepare the backup media according to the operating system documentation.
2. Verify that the database is not in use.

Use the following command to verify that the database is not in use:

```
proutil devel -C BUSY
```

3. Shut down `devel.db`.

   Enter the following command:

   ```
   proshut devel
   ```

4. Run PROBKUP `-estimate` to determine how much media is necessary for the backup, since this is the first time you are making a backup of the database.

   Use the following command

   ```
   probkup devel a:\devback -com -red 5 -scan -estimate
   ```

   The following message tells you about the state of your system, and how much media is necessary for backup:

   ```
   devel requires a total of 338 blocks of full backup media.
   devel requires a total of 3 blocks of incremental backup media.
   devel requires a total of 14 blocks of incremental backup media
        with one overlap.
   Total number of allocated blocks = 336
   Total number of allocated blocks that are free: 2
   Total number of allocated blocks that are used: 334
   ```

5. Perform a full offline database backup.

   - Enter the following command to perform a full database backup:

     ```
     probkup devel a:\devback -vs 708 -verbose -com -red 5 -scan
     ```

   - Enter the following command to perform a full online database backup:

     ```
     probkup online devel a:\devback -verbose -com -red 5
     ```

     Note: You cannot use the `-scan` parameter for online backups.

   The following parameters are utilized in the preceding commands:

   `devel`

   Identifies the name of the database you are backing up.

`online`

Specifies that the backup is an online backup.

`a:\devback`

Specifies the output destination is a file, `\devback`, on the `a:` drive.

`-verbose`

Tells PROBKUP to display information at 10-second intervals during the backup.

`-com`

Indicates that the data should be compressed before it is written to the disk drive. If you specify the `-com` parameter and do not use `-scan`, PROBKUP displays the number of blocks and the amount of backup required for an uncompressed database.

`-red 5`

Creates one error-correction block for every five blocks that are backed up.

`-scan`

Allows the backup utility to scan the database before backing it up to determine the number of blocks to be backed up.

As the full **offline** backup of `devel.db` runs, the following report appears:

```
64 bi blocks will be dumped.
336 out of 336 blocks in devel will be dumped.
This will require 369664 bytes of backup media.
This backup will require a minimum of 400 blocks to restore.
1 volume will be required.
Backed up 400 blocks in 00:00:04.
Wrote a total of 12 backup blocks using 369664 bytes of media.
Backup complete.
```

As the full **online** backup of `devel.db` runs, the following report appears:

```
64 bi blocks will be dumped.
336 out of 336 blocks in devel will be dumped.
This will require 369664 bytes of backup media.
This backup will require a minimum of 400 blocks to restore.
Backed up 400 blocks in 00:00:04.
Wrote a total of 12 backup blocks using 369664 bytes of media.
Backup complete.
```

The number of backup blocks is the number of `-bf` units written to the tape. Backup blocks contain data, primary recovery (BI), and error-correction blocks.

This example backs up a very small database. Using the `-red` parameter on a larger database increases the amount of time and backup media required for the backup.

6. If you have after-imaging enabled, back up the AI files onto a separate disk using a separate operating system backup utility.

Verifying a backup

Immediately after backing up the database, verify that the backup does not contain any corrupted blocks. Use the Restore (PROREST) utility to verify the integrity of a full or incremental backup of a database as follows:

- Run PROREST with the Partial Verify (`-vp`) parameter. With this parameter, PROREST checks the backup for bad blocks and reports whether any exist.
- Run PROREST with the Full Verify (`-vf`) parameter. With this parameter, PROREST compares the backup to the database block-for-block.

These parameters do not actually restore the database. They only verify the status of the backup, notify you if there are any bad blocks, and report whether the blocks are recoverable. You must run the restore utility again (without the partial or full verify parameters) to restore the database.

When you use the `-vp` parameter, PROREST scans the backup and recalculates the CRC code for each block. It then compares the newly calculated CRC code with the CRC code stored in the block header. If the codes do not match, PROREST marks the block as bad and displays the following message:

```
CRC check failed reading backup block n
```

If the backup contains error-correction blocks and a redundancy set contains only one bad block, PROREST uses the error-correction block (and the other blocks in the redundancy set) to re-create the bad block. The error-correction block is the EXCLUSIVE OR of the backup blocks in the redundancy set. When PROREST recovers the block, the following message appears:

```
Recovered backup block n
```

If the redundancy set contains more than one bad block or if the backup does not include error-correction blocks, PROREST cannot recover the bad block and displays the following message:

```
n CRC error within recovery group - recovery impossible
```

PROREST also cannot recover a corrupted block if the error-correction block itself has a CRC check failure. In this case, the following message appears:

```
Unable to recover previous block in error
```

If PROREST encounters 10 unrecoverable errors during the verify pass or during the database restore, you can terminate the verify operation, as shown:

```
10 read errors have occurred.
Do you want to continue? [y/n]
```

CRC codes and redundancy in backup recovery

To recover corrupted backup blocks, PROREST relies on:

- CRC codes to identify bad blocks. A CRC code is automatically calculated for each database backup block whether or not you specify a redundancy factor.
- Error-correction blocks to recover bad blocks. Error-correction blocks are included in the backup only if you explicitly request them with the `-red` parameter of the backup utility.

CRC codes

When PROBKUP writes a block of data to the backup media, it calculates a CRC code based on the contents of the block and stores it with the block. When restoring, PROREST re-examines the contents of the block and verifies that they are consistent with the accompanying CRC code. If the block contents are not consistent with the CRC code, the backup block is corrupted.

If the backup includes error-correction blocks, PROREST automatically uses the information in those blocks to recover the corrupted block. If the backup does not include error-correction blocks, PROREST cannot recover the corrupted block when you restore the database.

Error-correction blocks

Error-correction blocks contain information about the preceding set of backup blocks and allow PROREST to recover corrupted blocks in a backup. The error-correction block and the blocks it is based on are called a *redundancy set*. You can provide error-correction blocks in the backup by specifying the `-red` parameter in the backup command.

The `-red` parameter specifies a redundancy factor. The redundancy factor determines how many backup blocks are in each redundancy set. For example, if you specify a redundancy factor of 2, PROBKUP creates an error-correction block for every two backup blocks. Therefore, every redundancy set contains two backup blocks and an error-correction block.

PROREST can recover a bad backup block if it is the only corrupted block in the redundancy set. If a redundancy set contains more than one bad backup block or a bad backup block and a bad error-correction block, PROREST cannot recover any of the bad blocks in the redundancy set.

If you specify a very low redundancy factor (for example, 2), the chance of having two or more bad database blocks in a redundancy set is low. If you specify a higher redundancy factor, the chances are higher. However, lower redundancy values also produce larger backups that require more time and media. If the backup media is highly reliable, you might use a high redundancy factor; if the media is less reliable, you might want to specify a lower redundancy factor.

The size of each backup block—and therefore of each error-correction block—is determined by the `-bf` parameter. The default blocking factor is 34. For example, if the database block is 1,024 bytes and the blocking factor is 40, each backup block is 40K; that is, the size of 40 database blocks.

Restoring a database

In the event of database loss or corruption, you can restore the database from a backup. You must restore a database with the same version of OpenEdge that you used to create the backup.

This section describes:

- Using the PROREST utility to restore a database on page 112

- Important rules for restoring backups on page 113
- Obtaining storage area descriptions using PROREST on page 113
- Database restore examples on page 114

Using the PROREST utility to restore a database

Use the PROREST utility to restore a full or incremental backup of a database:

```
prorest dbnamedevice-name{ -list | -vp | -vf }
```

dbname

Specifies the name of the database where you want to restore the backups.

device-name

Identifies the directory pathname of the input device or standard file from which you are restoring the data.

-list

Provides a description of all application data storage areas contained within a database backup. Use the information to create a new structure description file and database so you can restore the backup. For additional information, see Obtaining storage area descriptions using PROREST on page 113.

-vp

Specifies that the restore utility reads the backup volumes and computes and compares the backup block cyclical redundancy checks (CRCs) with those in the block headers.

To recover any data from a bad block, you must have specified a redundancy factor when you performed the database backup. See Error-correction blocks on page 117 for more information about error-correction blocks and data recovery.

-vf

Specifies that the restore utility compares the backup to the database block-for-block. Do not compare the backup to a database that is in use.

Note: When you specify the -vp or -vf parameter, PROREST does not actually restore the database. You must restore the database in a separate step.

The first time you start the database after restoring an online backup, normal crash recovery runs and any transactions that were incomplete at the time of the backup are discarded.

When you restore a full database backup, consider restoring the backup to a new database. This allows you access to the corrupted database, if necessary. You must restore an incremental database backup to a restored database.

If PROREST encounters corrupted backup blocks that it is unable to recover, you lose the data in the corrupted blocks. The amount of lost data is approximately equal to the number of bad blocks multiplied by the blocking factor.

As you begin the restore procedure for a database, a report appears that indicates the date of the backup and the number of blocks required to restore the database.

Important rules for restoring backups

There are several important rules you must follow when you are restoring an incremental backup:

- If you are restoring over an existing database, verify your backup before executing the restore. If the existing database is the only copy, back up the existing database first.
- Restore a backup with the same OpenEdge release that you used to perform the backup. In most cases, you can restore a backup with any minor release within the same major release, with the following exceptions:
  - If you enabled a feature, you cannot restore your database to a release that does not support that feature, without first disabling the feature from your database. For example, if your database has Change Data Capture enabled, you cannot restore a backup of that database to an OpenEdge release that does not support Change Data Capture. Be aware that New VST tables are enabled by default starting in Release 11.5, and are a feature under this restriction. Use PROUTIL DESCRIBE to list the features enabled for your database.
  - For online backup ONLY, if you do not specify “bibackup all”, online backup is not backward compatible to releases prior to OpenEdge Release 11.3.0. The “bibackup” qualifier indicates whether to backup the active BI clusters or all the BI clusters. By default, only the active clusters are backed up.
- You must restore an incremental database backup to an existing database.
- Create a void database before you restore the backup, or else use the existing structure, overwriting it.
- You must restore a database in the same order that you backed it up. For example, you must first restore the full backup, followed by the first incremental backup, followed by the second incremental backup, etc. If you try to restore a database out of sequence, you get an error message and the restore operation fails.
- If you lose the second incremental and you used an overlap factor of 1, the third incremental correctly restores the data lost in the second incremental.
- After you restore a full backup, do not use the database if you want to restore successive incremental backups. If you make any database changes before completely restoring all backups, any successive, incremental backups (that were not restored) are rejected unless you restart the restore procedure beginning with the full backup.
- If a system failure occurs while you are restoring the database, restart the restore operation beginning with the backup volume that you were restoring at the time of the system failure.
- If a target database exists, it must have the same block size and storage area configuration as the source database. The PROREST utility attempts to expand storage areas to allow a complete restore, but if the storage areas cannot expand, the restore fails.

Obtaining storage area descriptions using PROREST

Use the PROREST utility with the `-list` parameter to obtain a description of the application data storage areas within a database backup. For example:

```
prorest db-name device-name -list
```

The following example shows the output from the `prorest -list` command:

```
OpenEdge Release 10.0A as of Fri Nov 7 02:56:17 EST 2003
Area Name: Schema Area
       Size: 7680, Records/Block: 32, Area Number: 6, Cluster Size: 1
Area Name: Info Area
       Size: 1024, Records/Block: 32, Area Number: 7, Cluster Size: 1
Area Name: Customer/Order Area
       Size: 2560, Records/Block: 32, Area Number: 8, Cluster Size: 8
Area Name: Primary Index Area
       Size: 32, Records/Block: 1, Area Number: 9, Cluster Size: 8
Area Name: Customer Index Area
       Size: 256, Records/Block: 1, Area Number: 10, Cluster Size 64
Area Name: Order Index Area
       Size: 8192, Records/Block: 32, Area Number: 11, Cluster Size 64
```

Use this information to create a new structure description (`.st`) file and database so you can restore the backup. If you do not create a structure description file, PROREST creates one variable-length extent for each storage area.

Database restore examples

This section includes examples of database restores using PROREST. The examples use UNIX syntax; for Windows, the only difference is the format of drive specification.

Full backup restore example

The database administrator of Company X's Development department wants to restore the `devel.db` database that was previously backed up.

To restore devel.db to a new database from a full backup:

1. Use PROREST.

   Enter the following command:

   ```
   prorest newdev /dev/rrm/0m
   ```

   The `newdev.db` database is an empty database. The 9-track tape drive (`/dev/rrm/0m`) specifies the device from which the full backup is being restored.

   As the restore begins, the following report appears:

   ```
   This is a full backup of /usr1/develop/devel.db. (6759)
   This backup was taken Wed Nov 18 15:34:43 1999. (6760)
   The blocksize is 1024. (6990)
   It will require a minimum of 3065 blocks to restore. (6763)
   Read 41 blocks in 00:00:02
   ```

   This command restores the database `devel.db` from a tape to `newdev.db`. The report indicates that volume 1 is being processed.

2. Connect to `newdev.db` once the restore is complete.

Incremental backup restore example

If you want to restore an incremental backup of the `devel.db` database to a new database, you must first restore a full backup.

To restore an incremental backup of `devel.db` to a new database:

1. Use PROREST to run an incremental restore of the database from a tape once the full restore is done.

   The syntax for the incremental restore is:

   ```
   prorest newdev /dev/rrm/0m
   ```

   The following report appears as the restore begins:

   ```
   This is an incremental backup of /usr1/develop/devel.db. (6759)
   This backup was taken Wed Nov 18 15:41:47 1999. (6760)
   The blocksize is 1024. (6990)
   It is based on the full backup of Wed Nov 18 15:34:43 1999. (6761)
   It will require a minimum of 3065 blocks to restore. (6763)
   Read 41 blocks in 00:00:00
   ```

2. Once the restore is complete, connect to `newdev.db`.

6

Recovering a Database

This chapter explains the different ways to recover your OpenEdge database and transactions if your system or disks fail.

For details, see the following topics:

- Introduction to recovery mechanisms
- File locations that ensure safe recovery
- Developing a recovery plan
- Sample recovery plans
- After-imaging and roll-forward recovery commands
- Recovering from system failures
- Recovering from media failures
- Recovering from a full disk
- Truncating the BI file
- Releasing shared memory
- Recovering from a lost or damaged control area
- Unlocking damaged databases
- Dumping tables from a damaged database
- Forcing access to a damaged database

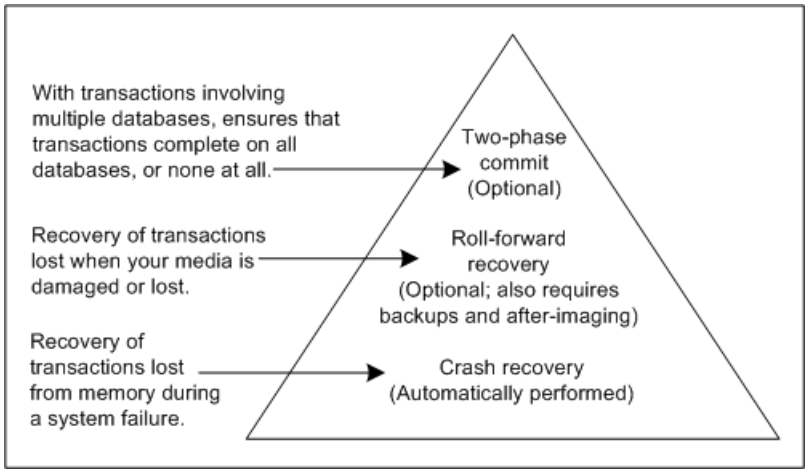
Introduction to recovery mechanisms

The OpenEdge RDBMS has three kinds of recovery mechanisms:

- **Crash recovery** — Uses primary recovery (BI) data to recover from system failures
- **Roll-forward recovery** — Uses backups and after-image data to recover from media failures
- **Two-phase commit** — Ensures that transactions occur consistently across multiple databases

Depending on your site requirements, you might choose not to implement all three of these recovery mechanisms. The following figure shows the order of precedence of these mechanisms. Crash recovery requires use of a recovery (BI) log and occurs without any interaction. Roll-forward recovery requires use of an after-image (AI) log. Two-phase commit requires use of a transaction log (TL). If you use two-phase commit, be sure to also use after-imaging.

Figure 5: OpenEdge recovery mechanisms



Each mechanism relies on notes that are written to a file to record database changes. A note is a record of the smallest unit of change in a database. For example, a record of one change made to one block in the database. The database engine automatically records database changes as notes in the primary recovery (BI) log. If after-imaging is enabled, it also records notes to the after-image (AI) log. If two-phase commit is enabled, it also records transactions and notes to the transaction log (TL).

Crash recovery

Crash recovery occurs automatically. With this feature, the database engine uses information from the primary recovery (BI) log to recover from system failures.

The BI files are a vital part of the database. You should treat the files as an integral part of the database unit. When you back up and restore the database, back up and restore the DB and BI files together. Never manually delete the BI files.

While the database is running, database transaction information exists in three areas:

- The database on disk
- The buffer pool in memory
- The BI files on disk

When database records are modified, the changes occur first in memory. When a transaction is committed, the change is recorded to the BI file. Over time, the database engine makes the change to the database file on disk. If the system fails, the information stored in the buffer pool is lost. The database engine performs crash recovery using the information logged to the BI file to re-create lost transactions and undo transactions that were not committed.

Before updating the database, the database engine makes a copy of the current information and writes it to the BI file. This activity begins at the end of an update operation. If the system fails during the transaction, the engine uses the information in the BI file to restore the database to its pretransaction state. The engine also uses the information in the BI files during normal processing to undo transactions.

For example, suppose you run the following ABL (Advanced Business Language) procedure:

```
FOR EACH customer:
  UPDATE customer.name customer.max-credit.
END.
```

You update customers 1 and 2, and while you are updating customer 3, the system fails. When you restart the database, messages appear in the database `.lg` file, similar to the following (be aware that the timestamp, PID, and thread ID is removed from the beginning of the line to facilliate easier reading of the messages):

```
I DBUTIL : (18186) Crash Recovery Status Interval (-crStatus): 5 seconds
I DBUTIL : (18187) Crash Recovery Transaction Display (-crTXDisplay): ON
I DBUTIL : (5326)  Begin Physical Redo Phase at 64 .
I DBUTIL : (7161)  Physical Redo Phase Completed at blk 4 off 718 upd 226.
I DBUTIL : (13547) At end of Physical redo, transaction table size is 128.
I DBUTIL : (12080) Begin Replication Redo Phase for 5 live transactions from 64 .
I DBUTIL : (12081) Replication Redo Phase completed at block 1800, offset 7328.
I DBUTIL : (7163)  Begin Physical Undo 6 transactions at block 4 offset 754
I DBUTIL : (5331)  Physical Undo Phase Completed at 220 .
I DBUTIL : (7162)  Begin Logical Undo Phase, 5 incomplete transactions are being backed out.
I DBUTIL : (11231) Logical Undo Phase begin at Block 220 Offset 116.
I DBUTIL : (12095) Logical Undo Phase completed at block 29568, offset 450.
I DBUTIL : (7129)  Usr 0 set name to shannon.
I DBUTIL : (123)   .bi file truncated.
I DBUTIL : (15109) At Database close the number of live transactions is 0.
I DBUTIL : (15743) Before Image Log Completion at Block 56773 Offset 3841.
```

The messages indicate the necessary phases of crash recovery performed by the database engine to bring the database to the consistent state that existed prior to the system failure. Since the engine performs crash recovery every time you open the database, not all of the recovery phases are logged in the database `.lg` file. For example, the engine performs and logs the Physical Redo phase unconditionally, but the Physical Undo and Logical Undo phases are only performed and logged when outstanding transactions are found. Similarly, the Replication Redo phase is only for a replication-enabled database.

When you rerun the same procedure, customers 1 and 2 remain updated, but the database engine has used the BI file to restore customer 3 to the state it was in before you began the update.

With a larger database with more activity, crash recovery can take longer than a few seconds to run. Two startup parameters can help you monitor the progress of your crash recovery by logging status messages to the screen:

- **`crStatus`** — Displays the phase progress as an estimate of percent complete. For example, the following is displayed to the screen during crash recovery with `-crStatus 5`:

```
Logical Undo  9% at block 2967. 2926 blocks processed in 5 seconds.
Logical Undo 19% at block 5744. 5653 blocks processed in 10 seconds.
Logical Undo 29% at block 8916. 8881 blocks processed in 15 seconds.
Logical Undo 38% at block 11543. 11502 blocks processed in 20 seconds.
Logical Undo 48% at block 14294. 14255 blocks processed in 25 seconds.
Logical Undo 56% at block 16850. 16819 blocks processed in 31 seconds.
Logical Undo 65% at block 19545. 19500 blocks processed in 36 seconds.
Logical Undo 75% at block 22587. 22474 blocks processed in 41 seconds.
Logical Undo 84% at block 25138. 25043 blocks processed in 46 seconds.
Logical Undo 93% at block 27725. 27704 blocks processed in 51 seconds.
Logical Undo 100% at block 29568. 29573 blocks processed in 54 seconds.
```

  For more information about crash recovery progress messages, see Crash Recovery Status (-crStatus) on page 497.

- **`crTXDisplay`** — Displays transaction IDs and adds counters to the `-crstatus` output. For example:

```
Transaction 2048 by user 15 at 2015/08/06@16:53:49. Counter: 202
Transaction 5771 by user 12 at 2015/08/06@16:55:15. Counter: 459
Transaction 24 by user 7 at 2015/08/06@16:52:49. Counter: 1
Transaction 5801 by user 10 at 2015/08/06@16:55:15. Counter: 460
Transaction 5802 by user 14 at 2015/08/06@16:55:16. Counter: 463
Transaction 5683 by user 18 at 2015/08/06@16:55:15. Counter: 459
Logical Undo 10% at block 3205. 3200 blocks processed in 5 seconds. Counter: 413
Logical Undo 21% at block 6427. 6378 blocks processed in 10 seconds. Counter: 363
Logical Undo 30% at block 9038. 9015 blocks processed in 15 seconds. Counter: 322
Logical Undo 38% at block 11581. 11464 blocks processed in 20 seconds. Counter: 283
Logical Undo 44% at block 13148. 13097 blocks processed in 25 seconds. Counter: 258
Logical Undo 50% at block 14994. 14963 blocks processed in 31 seconds. Counter: 229
Logical Undo 58% at block 17502. 17447 blocks processed in 36 seconds. Counter: 190
Logical Undo 64% at block 19210. 19195 blocks processed in 41 seconds. Counter: 163
Logical Undo 71% at block 21264. 21237 blocks processed in 46 seconds. Counter: 131
Logical Undo 82% at block 24412. 24361 blocks processed in 51 seconds. Counter: 82
Logical Undo 90% at block 27070. 26951 blocks processed in 56 seconds. Counter: 41
Logical Undo 100% at block 29568. 29573 blocks processed in 60 seconds. Counter: 1
```

  For more information, see Crash Recovery Transaction Display (-crTXDisplay) on page 498.

Crash recovery protects you from system failures, but it does not protect you from loss of media. In the event of media loss, you must restore from a backup, and either manually re-enter the lost transactions or use the roll-forward recovery mechanism to re-create the transaction.

Roll-forward recovery

Roll-forward recovery, used together with after-imaging and backup, lets you recover from media failures. When a database disk fails, you can restore the most recent backup, then use roll-forward recovery to restore the database to the condition it was in before you lost the disk. With roll-forward recovery, the database engine uses data in the AI files to automatically reprocess all the transactions that have been executed since the last backup was made.

To use roll-forward recovery, you must:

- Perform regularly scheduled backups of the database. Regular backups are a fundamental part of recovery.
- Enable after-imaging immediately after you complete the backup. See After-imaging on page 139 for information about enabling after-imaging.
- Perform regularly scheduled backups of the AI files. See Backing Up a Database on page 97for information about backing up the AI files.
- Store the AI files on different disks than those containing the database and BI files.

  When you enable after-imaging, the database engine writes database changes to the AI files. If you store the AI files on the same disks as the database or BI files and a disk is corrupted, you cannot use the AI files to recover that database.
- Archive the AI files to tape or other durable media as they become full.

This example shows how the database engine uses the AI files to restore the database. Suppose you run the following ABL procedure:

```
FOR EACH customer:
  UPDATE customer.name customer.max-credit.
END.
```

You update customers 1 and 2, and while you are updating customer 3, the disk where the database file is stored is damaged. You cannot use the BI file to restore the transactions because the original database is no longer valid.

However, because you enabled after-imaging, you can use roll-forward recovery to recover the database. If you do not enable after-imaging, you lose all the updates since the last database backup.

Before updating the database, the database engine makes a copy of the current information and writes it to the BI file and the AI file.

After updating customers 1 and 2, the database disk fails while updating customer 3. The AI files have a copy of all transactions completed since the last backup. Restore the last backup of the database and then roll forward the AI files to produce a restored database that contains all completed transactions.

Two-phase commit

Two-phase commit ensures that distributed transactions (that is, transactions involving multiple databases) occur consistently across all databases. Two-phase commit is not necessary for transactions involving a single database.

For detailed information on two-phase commit, see Distributed Transaction Processing on page 255.

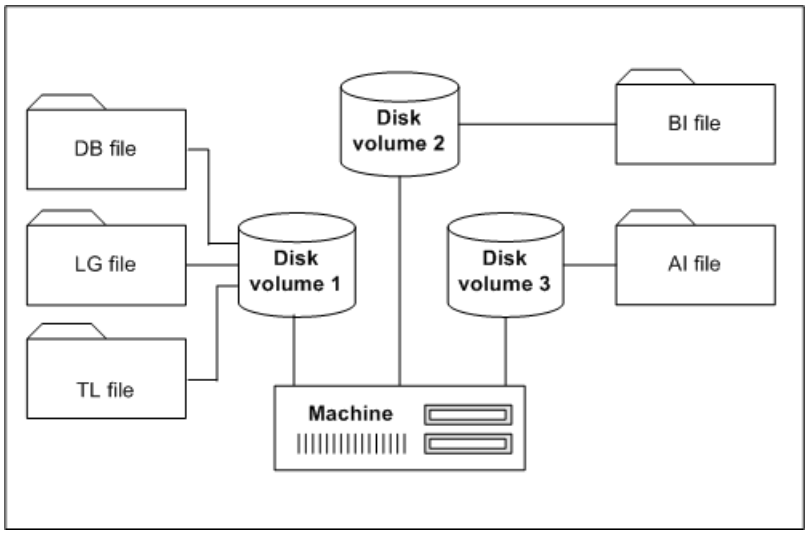
Note: If you use two-phase commit, you should also use after-imaging to ensure database integrity and avoid backup synchronization problems. Although two-phase commit ensures that two databases remain synchronized, it does not protect you from a lost database or BI file. If you lose an entire file or disk, you must use the AI file to roll forward to the point of the crash.

File locations that ensure safe recovery

The AI files, together with the most recent database backup, contain the same information as the DB files. If you lose the disk containing theDB and BI files, you can use the AI files and database backup to reconstruct the database. For this reason, you should place the AI files on a different disk from the one that contains the DB and BI files.

The following figure shows sample locations for each of the DB, BI, AI, and event log files.

Figure 6: Storing database files



Developing a recovery plan

A recovery plan documents how you will recover your system if it fails. When developing a recovery plan, you must consider every potential failure, from short-term power outages and disk failures to environmental disasters such as earthquakes.

To develop a database recovery plan, you first need to determine the availability requirements for the database application. Consider the following questions:

- How many transactions can you afford to lose?
- How long can the application be offline while you perform scheduled maintenance, such as backups?
- If the system or database becomes unavailable, how much time can you spend recovering?
- If you use transactions that affect more than one database, can you allow transactions to occur inconsistently in those databases?
- How will you test your recovery plan?

Use the tables in the following sections to develop a recovery plan. These tables provide a range of answers to these questions, and also provide backup and recovery suggestions for each.

Time needed for recovery

How you archive and restore data depends on how long you can spend recovering the database if it becomes unavailable. In addition to the time required to recover the database, you must factor in the time to repair hardware, file systems, system disks, and other system components.

Recovery guidelines

These are the guidelines to follow to ensure a safe recovery:

- **Always:**
  - Include step-by-step instructions and checklists in the recovery plan
  - Keep a hard copy of the recovery plan
  - Back up the database on a regular basis
  - Back up the AI files on different media from the database and BI files
  - Label, test, and keep the backups in a separate and secure location
- **Do Not:**
  - Store a BI file on the same disk as an AI file
  - Erase a DB, BI, or AI file unless you are certain you no longer want the database, or unless you have a secure, complete backup
  - Copy a database file without also copying the BI files associated with that database
  - Restore a database file without restoring the corresponding BI file
  - Copy a database with an operating system utility while the database is in use without running `PROQUIET`

Caution: If you run your OpenEdge database with the Unreliable Buffered I/O (`-r`) parameter and the system fails because of a system crash or power failure, you cannot recover the database. If you run with the No Crash Protection (`-i`) parameter and the database fails for any reason, you cannot recover the database.

Sample recovery plans

This section provides sample recovery plans to meet a range of availability requirements, including low, moderate, moderately high, and high availability requirements.

Example 1: Low availability requirements

The database administrators at site A have determined the following availability requirements. The table below provides examples.

Table 20: Sample low availability requirements

Question	Answer	Recovery technique
How many committed transactions can you afford to lose?	1 week of transactions	Perform weekly backups.
How long can the application be offline while you perform scheduled maintenance?	8 hours per day	Perform weekly backups offline.

Question	Answer	Recovery technique
How long can you spend recovering if the database becomes unavailable?	24 hours	Keep a full backup and any number of incrementals or AI files.
Must distributed transactions occur consistently across databases?	No	Do not implement two-phase commit.

Given these requirements, the database administrators perform a full offline backup every Monday at 5 PM. Incrementals are not required because the site can afford to lose one week of transactions, and full backups are performed weekly. The backup is tested Tuesday at 5 PM using disk space reserved for that purpose.

Example 2: Moderate availability requirements

The database administrators at site B have determined the following availability requirements. The table below provides examples.

Table 21: Sample moderate availability requirements

Question	Answer	Recovery technique
How many committed transactions can you afford to lose?	1 day of transactions	Perform daily backups.
How long can the application be offline while you perform scheduled maintenance?	8 hours per week	Perform weekly offline backups with daily online full or incremental backups.
How long can you spend recovering if the database becomes unavailable?	8 hours	Keep a full backup and one incremental or AI file.
Must distributed transactions occur consistently across databases?	No	Do not implement two-phase commit.

Given these requirements, the database administrators perform a full offline backup every Saturday night. Because they can lose only one day of transactions, they supplement weekly backups with daily online incremental backups. Because they can only allow time to restore a single incremental, they perform incremental backups with an overlap factor of six. This means that each incremental backup saves changes that were saved in the last six incremental backups. The full backup is tested Saturday night, and after the incremental backups finish, they are tested using disk space reserved for that purpose.

Example 3: Moderate-to-high availability requirements

The database administrators at site C have determined the following availability requirements. The table below provides examples.

Table 22: Sample moderate-to-high availability requirements

Question	Answer	Recovery technique
How many committed transactions can you afford to lose?	A few minutes of transactions	Enable after-imaging.

Question	Answer	Recovery technique
How long can the application be offline while you perform scheduled maintenance?	Never	Perform online backups with after-image files.
How long can you spend recovering if the database becomes unavailable?	24 hours	Restore a full backup and a series of any number of incrementals or AI files.
Must distributed transactions occur consistently across databases?	Yes	Implement two-phase commit.

Given these requirements, the database administrators perform a full online backup every Saturday night. Because they can afford to lose only a few minutes of transactions, the site also enables after-imaging. They switch AI extents every day at 7 PM and archive the resulting full after-image extent. (Because they switch AI extents at scheduled intervals rather than waiting for an extent to fill, they must be sure to monitor AI extent usage in case an extent fills before the scheduled switch.) Because applications at this site perform distributed transactions that must be consistent across databases, this site implements two-phase commit. The full backup is tested Saturday night. After they are archived, AI files are applied to the tested backup since so few transactions can be lost.

Example 4: high availability requirements

The database administrators at site D have determined the following availability requirements. The table below provides examples.

Table 23: Sample high availability requirements

Availability question	Answer	Recovery technique
How many committed transactions can you afford to lose?	None	Enable after-imaging.
How long can the application be offline while you perform scheduled maintenance?	Never	Perform online backups with after-image files.
How long can you spend recovering if the database becomes unavailable?	4 hours	Keep a duplicate system on warm standby. Use roll-forward recovery to keep the standby database current with the production database.
Must distributed transactions occur consistently across databases?	No	Do not implement two-phase commit.

Given these high availability requirements, the database administrators keep a duplicate database on warm standby, and follow these steps:

1. Restore a backup of the production database to an empty database with a duplicate structure on a different system.
2. On the production database, enable after-imaging using AI files with fixed-length extents.

3. On the production database, whenever a fixed-length AI extent becomes full, copy it to the standby system and roll it forward on the standby database.
4. After bringing the standby database up to date, mark the full after-image extent on the production database as empty to make it available for reuse.

In addition, backups and AI files are constantly tested and verified on the standby database. The standby database is put through crash recovery, verified, and if possible, backed up before restoring online backups of the production database to the standby database.

Sample recovery scenarios

To show how the parts of a recovery plan fit together, this section presents an overview of the steps and commands you use to implement a database recovery plan.

The series of figures that follows show sample scenarios of a one-week period.

Figure 7: Recovery scenario—day 1

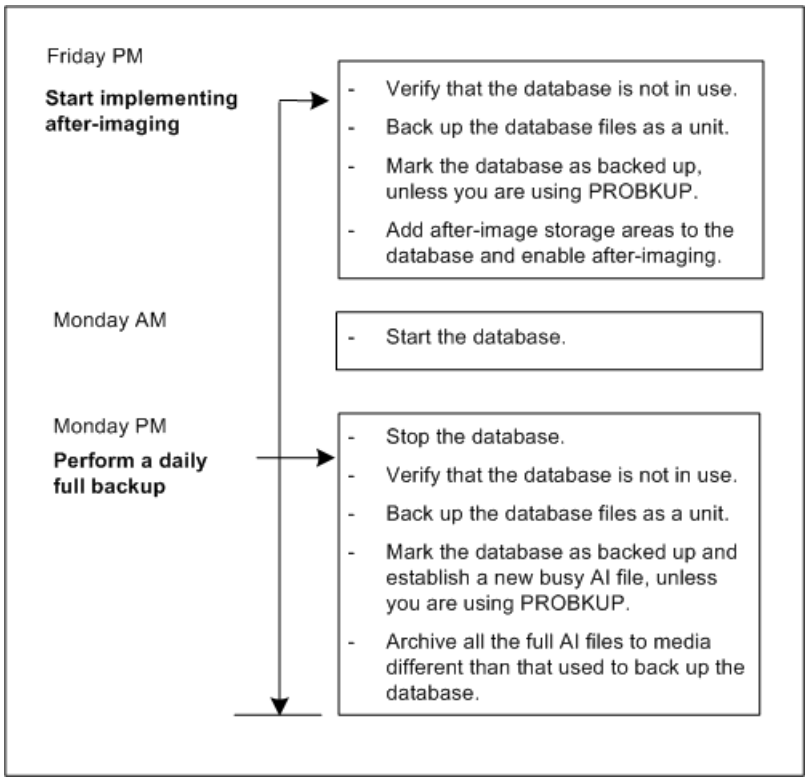


Figure 8: Recovery scenario—day 2

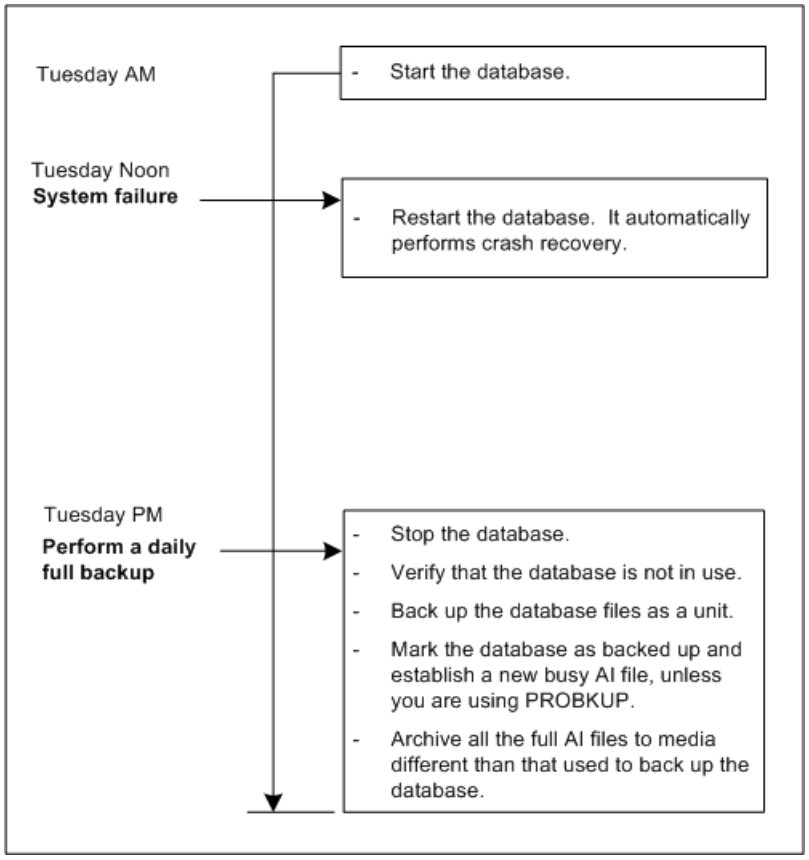


Figure 9: Recovery scenario—day 3

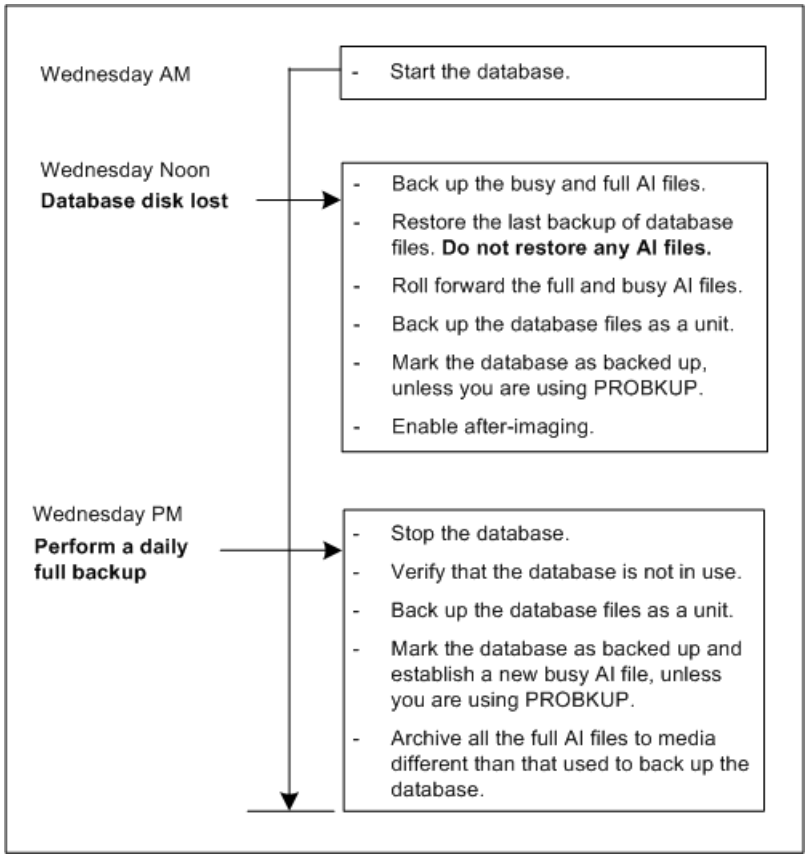


Figure 10: Recovery scenario—day 4

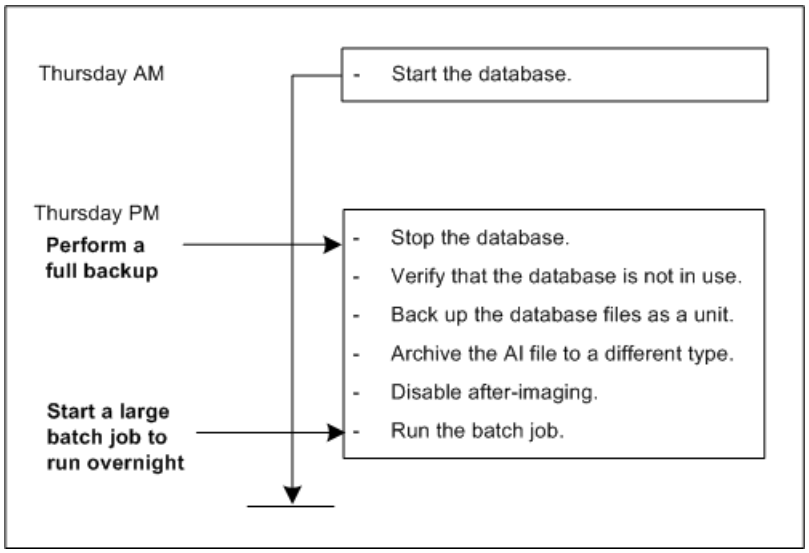
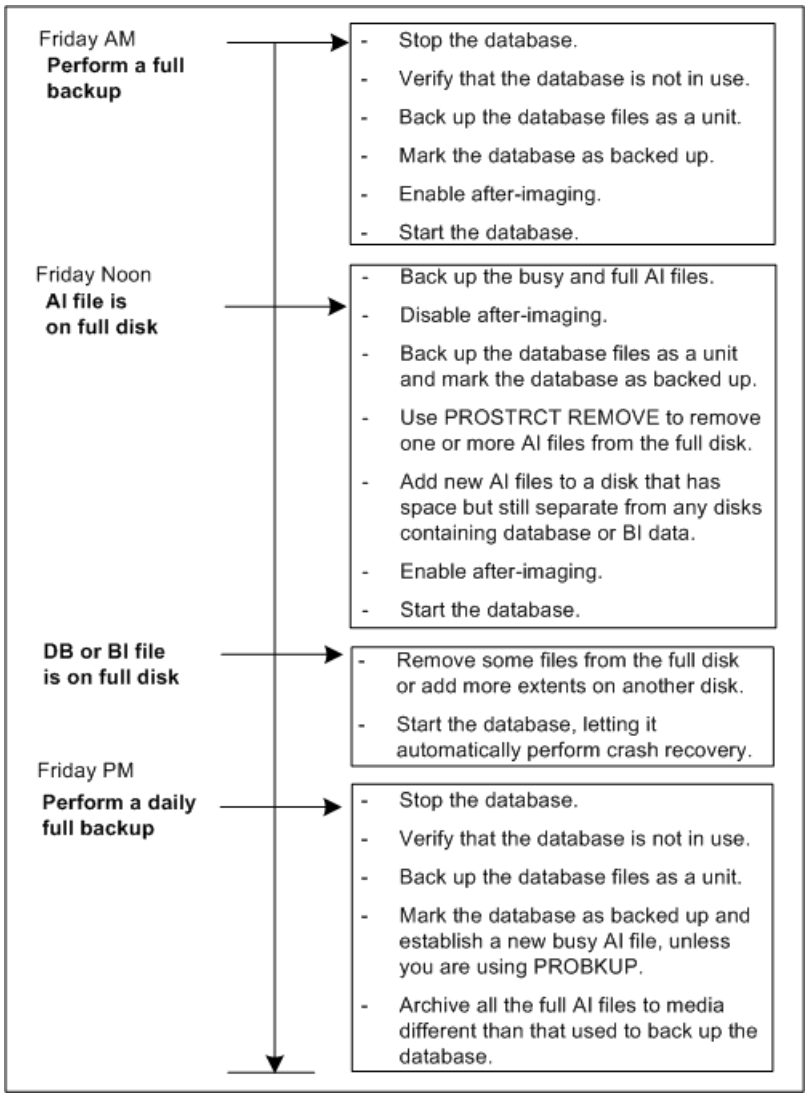


Figure 11: Recovery scenario—day 5



After-imaging and roll-forward recovery commands

The table below lists the utilities used to perform the steps in the scenarios shown in Sample recovery scenarios on page 126.

Table 24: Utilities used with roll-forward recovery

Utility command	Function
BUSY	Verifies that the database is not in use. The database cannot be in use when you perform a full offline backup.
PROBKUP or operating system utility	Backs up the database. You must perform a full backup of the database before you can use roll-forward recovery.

Utility command	Function
RFUTIL MARK BACKEDUP	Marks the database as backed up. If you do not use PROBKUP to back up the database, you must mark the database as backed up to begin after-imaging or start a new after-image file.
RFUTIL AIMAGE NEW	Starts a new AI file. After you back up the database and AI files, you can truncate the existing AI file.
RFUTIL AIMAGE END	Disables after-imaging. Before you run a batch job, you might want to disable after-imaging to improve performance.
RFUTIL ROLL FORWARD RETRY	Restarts the roll-forward operation. Use of this qualifier is limited to RFUTIL roll-forward operations that fail because of power outages or system failures.

Recovering from system failures

Before-imaging is automatically enabled to let you recover from a system failure. If the system fails, the technique for recovering the database depends on the database operation you were performing when the system crashed. The following sections provide specific information.

System crash while running RFUTIL ROLL FORWARD

If the system crashes while you are using the RFUTIL Roll Forward utility, use the RETRY qualifier to resume the roll forward:

```
rfutil db-name -C roll forward retry
```

Use of this qualifier is limited to RFUTIL roll-forward operations that fail because of power outages or system failures. The ROLL FORWARD RETRY qualifier restarts the roll-forward operation on the after-image extent that was in the process of rolling forward. The retry operation finds the transaction in process at the time of failure and resumes rolling forward.

System crash while running other utilities

If the system crashes while you are using any of the following utilities, simply rerun the utility:

- BUSY
- HOLDER
- TRUNCATE BI
- RFUTIL AIMAGE BEGIN
- RFUTIL AIMAGE END
- RFUTIL AIMAGE NEW

- RFUTIL MARK BACKEDUP
- RFUTIL ROLL FORWARD RETRY
- RFUTIL AIMAGE AIOFF
- RFUTIL AIMAGE END
- RFUTIL AIMAGE SCAN
- RFUTIL AIMAGE EXTENT EMPTY
- RFUTIL AIMAGE TRUNCATE
- RFUTIL AIMAGE EXTENT FULL

System crash while backing up the database

If the system crashes while you are backing up the database, restart the backup.

System crash while database is up

If the system crashes while the database is up, restart the database. Crash recovery automatically performs during startup using the information stored in the BI files.

Note: If you run the database with the `-r` parameter and your system fails because of a system crash or power failure, you cannot recover the database. If you run the database with the `-i` parameter and it fails for any reason, you cannot recover the database.

Recovering from media failures

After-imaging combined with backup and roll-forward recovery lets you recover the database if a disk holding a database file is damaged. After-imaging is optional; you must enable it before a disk fails, and you must have performed backups to use roll-forward recovery. You should also keep backups and archived AI files until they are no longer required.

The technique for recovering the database depends on the circumstances that require recovery. The next sections provide scenarios and explanations of how to use the PROUTIL and RFUTIL utilities to handle the following problems:

- Loss of the DB files, BI files, or both
- Loss of the AI files
- Loss of the database backup
- Loss of the TL file

Loss of the DB files, BI files, or both

If a disk holding the DB or BI files is damaged, or the files are accidentally deleted, you can restore the database.

To restore the database after the loss of DB and/or BI files:

1. Back up the current AIfile and any full AI files not already archived before rolling forward. This step is important to protect yourself in case you lose the AI files if anything goes wrong, such as inadvertently overwriting the AI file. Be sure to back up the AI files on media different from the media where the database files are backed up. The AI files have the information you require to bring the database backup files up to date.
2. Create the structure for the database. You will restore the database to this structure.
3. Restore the most recent full database backup. If you are using incremental backups, you must restore the most recent full backup and then apply all the incremental backups.
4. If you use after-imaging, roll forward the AI files starting after the last backup. Use the RFUTIL ROLL FORWARD utility. See After-imaging on page 139 for more information.
5. Perform a full backup of the restored, recovered database. Follow the standard backup procedure described in Backing Up a Database on page 97.
6. Restart after-imaging and two-phase commit, if you are using it.
7. Restart the database.
8. Restart the applications and continue database processing.

Loss of the AI file

The following procedure describes what to do if a disk holding the AI files is damaged or the AI file is accidentally deleted.

To recover from the loss of an AI file:

1. If two-phase commit is enabled for the database, end two-phase commit using the PROUTIL 2PHASE END utility. You must do so before you can perform the next step.

   Note: If you end two-phase commit and a coordinator recorded any distributed transaction commits in this database, the user cannot resolve any limbo transactions where this database was the coordinator.

2. Disable after-imaging using the RFUTIL AIMAGE END utility.
3. If you have a database, re-create the lost extent using PROSTRCT ADD.
4. Back up the database using the PROBKUP utility or an operating system backup utility.
5. If you used an operating system backup utility to back up the database, mark the database as backed up using the RFUTIL MARK BACKEDUP utility.
6. Enable after-imaging.
7. Re-enable two-phase commit if you disabled it in Step 1.

Loss of database backup

If after-imaging is enabled for a database, you can recover from a lost or corrupted database backup as long as you have archived the AI files every time you backed up the database.

To restore the database from archived AI files:

1. Archive the current AI file to ensure that a second copy is available in case the original extents are damaged (for example, if you inadvertently overwrite the current busy AI file when restoring archived AI files in Step 3). The AI files have the information you need to bring the database backup files up to date.

Be sure to archive the AI files on different media than the database files backup.

2. Restore the backup you made immediately before the lost backup. If you are using incremental backups, you must restore the most recent full backup and then apply all the incremental backups to the full backup.
3. Restore the AI files you archived at the same time you made the backup that was lost or corrupted.
4. Use the RFUTIL ROLL FORWARD utility to roll forward the AI files you restored in Step 3. See After-imaging on page 139 for more information.
5. Use the RFUTIL ROLL FORWARD utility to roll forward the current AI files (that is, the AI files in use since the lost backup).
6. Back up the database following the standard backup procedure described in Backing Up a Database on page 97.
7. Enable after-imaging and two-phase commit if you are using them.
8. Restart the database.
9. Restart any applications and continue database processing.

Loss of transaction log file

If you cannot access the database because of a lost or corrupt transaction log file, you must end and then enable two-phase commit.

To recover from the loss of a transaction log file:

1. If two-phase commit is enabled for the database, end two-phase commit using the PROUTIL 2PHASE END utility.

   Note: If you end two-phase commit and a coordinator recorded any distributed transaction commits in this database, the user cannot resolve any limbo transactions for that distributed transaction unless after-imaging is also enabled for the database.

2. Enable two-phase commit using the PROUTIL 2PHASE BEGIN utility.
3. Put the database back online.

Recovering from a full disk

The database engine terminates if a disk holding a database file becomes full while the database is running.

To recover from a full disk:

1. Remove any nonessential files from the full disk.
2. Start the database, letting crash recovery automatically perform.

If you cannot make enough room on your disk to allow the database engine to perform crash recovery, you must take further steps. The steps vary, depending on the contents of the full disk, as explained in the following sections.

After-image area disk

If a variable-length after-image extent fills a disk, the database engine automatically switches to the next AI extent if that extent is empty. If the extent is not empty, the database shuts down unless the After-image Stall (`-aistall`) startup parameter was specified.

To restart a database after a full AI extent causes a shutdown:

1. Back up the least recently used full after-image extent. Use the RFUTIL AIMAGE EXTENT FULL utility to find out which extent this is.
2. Use the RFUTIL AIMAGE EXTENT EMPTY utility to mark the extent as available for reuse.
3. Restart the database.

Control or primary recovery area disk

The procedure you follow to recover from a full disk depends on whether the variable-length BI extent is on the full disk. If the variable-length BI extent is on the full disk, you cannot start the database, because the BI extent grows during the crash recovery phase of the database startup.

BI file not on full disk

Use the following procedure to recover from a full disk if the BI file is not on the full disk:

1. Truncate the BI file using the PROUTIL TRUNCATE BI utility. This rolls back any transactions active at the time the database shuts down. You can also start a single-user OpenEdge session to accomplish this.
2. Add one or more data extents to the database using the PROSTRCT ADD utility. This step is not necessary to bring the database back online. However, if you omit this step, the database will shut down when users begin adding data to it.

   If you have no additional disk space, you must install a new disk or delete some data from the database before you can bring the database online.

3. Restart the database.

BI file on the full disk

Use the following procedure to recover from a full disk if the BI file is on the full disk:

1. Use the PROSTRCT ADD utility to add a BI extent on a disk that contains some free space. The maximum amount of additional BI space you require to recover the database is variable, and may exceed the current size of the database. If you have a disk with this much free space, then assign the new BI extent to that volume. Note that this is a worst-case space requirement and the BI file normally grows much less than this. Adding the BI extent makes the variable length extent a fixed-length extent.
2. Start a single-user OpenEdge session against the database. This begins database crash recovery. Exit the single-user session.
3. Restart the database.

Transaction log file disk

Recovering from a full disk when two-phase commit is enabled requires disabling two-phase commit.

To recover from a full disk if two-phase commit is enabled:

1. Use PROUTIL 2PHASE END to disable two-phase commit on the database.

   Note: Unless after-imaging is also enabled for the database when you end two-phase commit, the user will be unable to resolve any limbo transactions where the database recorded the coordinator's commit of a distributed transaction.

2. Truncate the BI file.
3. Enable two-phase commit.
4. Restart the database.

Truncating the BI file

Truncating the BI file resets the file to its minimum size. You should plan ahead to determine the amount of disk space required to hold all the files required to process at your peak transaction rate. However, under certain circumstances, the BI file might grow unusually large. This usually occurs because of a long-running transaction; for example, if a user leaves the terminal for a long period in the middle of a transaction. If a situation like this occurs, you can truncate the BI file.

When you truncate the BI file, the database engine uses the information in the BI files to bring the database and AI files up to date, waits to verify that the information has been successfully written to the disk, then truncates the primary recovery file to its minimum length.

To truncate the BI file, use the TRUNCATE BI qualifier of the PROUTIL utility:

```
proutil db-name -C truncate bi
```

In this command, *db-name* specifies the database you are using.

See PROUTIL TRUNCATE BI qualifier on page 776 for a complete description.

Releasing shared memory

If the broker dies or is killed by some means other than the database shutdown command, it does not release attached shared-memory segments. Use PROUTIL -C DBIPCS to verify shared memory has not been released; use the UNIX command `ipcrm -m` to free it.

Use PROUTIL with the DBIPCS qualifier to display the status of shared-memory segments attached by all OpenEdge databases on the system:

```
proutil -C DBIPCS
```

You do not have to include a database name.

Here is an example of the status output:

```
OpenEdge SHARED MEMORY STATUS
 ID  ShMemVer  Seg#  InUse   Database
 68      -      -     -      (not OpenEdge)
100      3      0     Yes    /db/work5/sports
101      3      1     -      /db/work5/sports
120      3      0     No     /db/work5/test
150      2      -     -      (unsupported shared memory version)
```

The table below describes the display fields in the output.

Table 25: Shared-memory segment status fields

Field	Description
ID	Indicates the shared-memory ID.
ShMemVer	Specifies the shared-memory version.
Seg#	Indicates the shared-memory segment number. One database can own more than one segment.
InUse	Specifies whether the segment is in use. Yes or No values are displayed only if the segment is number 0. All other segments show a dash (-). To determine whether a set of segments is in use, check the InUse value of segment 0 for the relevant database.
Database	Represents the full path name of the database.

Recovering from a lost or damaged control area

Use the PROSTRCT BUILDDB utility to recover when an existing database control area (DB) is lost or damaged. The utility re-creates a control area from the structure description (`.st`) file of an existing database. For example:

```
prostrct builddb db-name structure-description-file
```

PROSTRCT BUILDDB uses the following parameters:

db-name

Specifies the database you are using.

structure-description-file

Specifies the existing structure description file. If a structure description file is not specified, PROSTRCT BUILDDB assumes the name is *db-name*`.st`.

PROSTRCT BUILDDB does minimal validation of the resulting control area.

Unlocking damaged databases

With databases, inconsistencies can occur between extents from accidental misuse of operating system copy utilities or from incorrectly administered backup and restore procedures. The database engine synchronizes opening and updating the DB, BI, and AI extents to ensure the consistency of the database. If the engine finds an inconsistency among these extents, it returns an error message and stops any attempt to open the database. If this happens, try to restore the database from backups. As a last resort, you can use the PROSTRCT utility with the UNLOCK qualifier to open the database to dump the data. Use PROSTRCT UNLOCK only as a last resort. The locked database might not unlock, or the resulting unlocked database might be damaged and invalid.

To unlock a damaged database:

1. Unlock the database with PROSTRCT UNLOCK.

   The syntax is as follows:

   ```
   prostrct unlock db-name
   ```

   Note: If you are missing any database extents when you run PROSTRCT UNLOCK, PROSTRCT replaces any missing extents with empty formatted extents and displays a message. You can determine whether this has occurred the next time any utility or program tries to open the database.

2. Use the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface to dump the data definitions and the table contents.
3. Exit the database.
4. Create a new void database.
5. Restart the database and reload the dumped data definitions and data into the new database.

Dumping tables from a damaged database

Occasionally, data is damaged and the index rebuild utility does not work. When this happens, you can use the `p-dump.p` procedure to dump the data in a table called item. The `p-dump.p` procedure runs slowly on large databases. Use it only when you cannot use the Index Rebuild utility.

Table 26: p-dump.p

```
DEFINE VARIABLE ix AS INTEGER NO-UNDO.
FIND _file "item".
OUTPUT TO item.d.
DO ix = 1 TO 10000:
  FIND item WHERE RECID(item) = ix NO-ERROR.
  IF AVAILABLE item AND ix <> INTEGER(_file._template) THEN
    EXPORT item.
END.
```

In this procedure, data from the item table is dumped into the `item.d` file. Use the `item.d` file with the reload procedure. For more information on loading, see Dumping and Loading on page 411

In the p-dump.p procedure, you must set the end value for the DO block high enough (10,000 in the previous example procedure) so that every record in the database is examined. Calculate a safe value using the following formula:

```
100 + 32*(database-size-in-bytes / database-block-size)
```

The database block size varies among systems. Use the PROSTRCT STATISTICS utility to determine the database block size for your system.

Forcing access to a damaged database

If you are unable to regain access to your database, you should restore the database from backup. If you have no backup and no other alternative, you can use the -F startup parameter to force entry to a damaged database.

Caution: You should use the -F parameter only as a last resort, since it compromises the integrity of the database.

To force access to a damaged database:

1. Back up the database.
2. Truncate the BI files using the PROUTIL command with the -F parameter:

   The syntax is as follows:

   ```
   proutil db-name -C truncate bi -F
   ```

   In this command, *db-name* specifies the name of the database.

   The following messages appear:

   ```
   The -F option has been specified to . (6260)
   Forcing into the database skips database recovery. (6261)
   ```

3. Start a single-user session against the database.
4. Use any of the database administration tools or utilities to dump and then load the database. See Dumping and Loading on page 411 for more information.

   Caution:

   Forcing access to a damaged database is an action of last resort and might not result in successfully opening the database. If the database cannot open, none of the OpenEdge recovery tools can be used to dump the contents of the database.

7

After-imaging

The after-imaging feature lets you recover a database that was damaged when a failure caused the loss of the database or primary recovery (before image) area. When you enable after-imaging, the database engine writes notes containing a description of all database changes to the after-image (AI) files. You can use the AI files with the roll-forward recovery process to restore the database to the condition it was in before you lost the database, without losing completed transactions that occurred since the last backup.

For details, see the following topics:

- After-image areas and extents
- Estimating after-imaging disk space requirements
- Creating after-image areas
- Enabling after-imaging offline
- Enabling after-imaging online
- Managing after-imaging files
- AI File Management utility
- Add and reorder AI extents
- Performing roll-forward recovery
- After-image sequences
- Disabling after-imaging

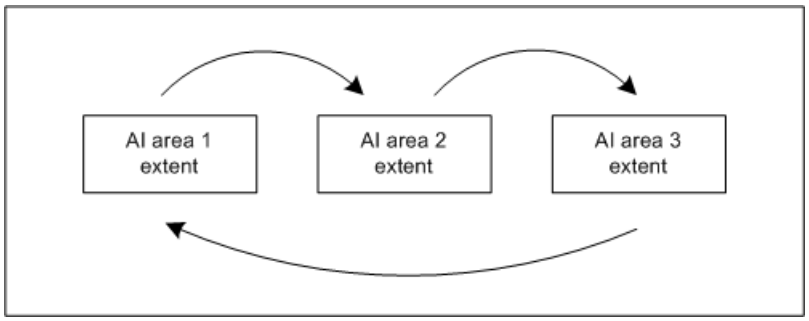
After-image areas and extents

You define AI areas in the structure description (`.st`) file. You can create a single AI area or multiple areas on a single disk or multiple disks. With multiple areas, you can perform online backups while after-imaging is enabled. Regardless of how many AI areas are defined, each AI area contains only one extent. Before defining the AI extents, consider the following:

- The database engine fills the AI areas in the order that you define them in the structure description file. When you are defining areas, you can store more than one AI area on a disk. However, you should store all the AI areas on disks other than the one that contains the database (DB) files or primary recovery area (BI) files.
- For both fixed-length and variable-length extents, the database engine automatically switches extents when the current extent becomes full, as long as the next extent is empty. If you define three large fixed-length extents, you can use extent 1 for a full day's worth of transactions, and have extent 2 empty and ready to use when you need to switch over to the next extent. This also leaves extent 3 available if you perform an unusually high number of transactions and use both extents 1 and 2.

The database engine uses AI areas sequentially, in the order defined in the structure description file. AI area filenames have a `.an` extension, where `n` indicates the numerical order in which you defined the area. After it uses the last area, the database engine reuses the first area if it is empty. The following figures illustrates this behavior. An *extent switch* is the operation of switching from one AI area extent to another.

Figure 12: After-image extents switching



You must monitor the status of the extents to ensure that you do not try to reuse an unavailable file. For information on monitoring the status of your AI extents, see Monitoring AI file status on page 144.

Like database and BI extents, there are two types of AI extents:

- Fixed-length extents on page 140
- Variable-length extents on page 140

Fixed-length extents

Fixed-length extents are extents that are preallocated and preformatted. With fixed-length extents, you control how much disk space each extent uses by specifying the size in the structure description file.

Variable-length extents

Variable-length AI extents do not have a predefined length. They continue to fill until they use the entire disk, you back up the database, or you issue the RFUTIL AIMAGE NEW command. The initial length of a variable-length extent is 128K. You can define more than one variable-length AI area for a single database.

Estimating after-imaging disk space requirements

Before creating after-image files, it is important to accurately estimate the amount of disk space required to hold all the AIdata for your database. Even when you use variable extents, if the disk runs out of space and there is no empty AI extent available, you must perform emergency maintenance or the database is forced to shut down. See Recovering a Database on page 117 for details. To prevent the database engine from shutting down when it exhausts AI disk space, start your database with the after-image stall (`-aistall`) startup parameter. When after-image stall is enabled, the database will suspend all activity, and send an error message to the log file indicating that the AI extent is full. For more information on after-image stall, see After-image Stall (-aistall) on page 485.

To determine the amount of AI file space required, use the PROMON utility's **BI Log Activity Display** option to monitor the number of BI writes that occur during a period of time. The information that is written in the BI file closely matches what is written in the AI file. Typically 0.05 percent fewer bytes are written to an AI file than are written to the BI file. Therefore, the **BI Bytes written** statistic provides a close estimate of the space required for an AI file for the same activity period.

Measure the **BI Bytes written** statistic several times toward the end of the activity period that you are sampling. If your workload varies from day to day or peaks during end-of-month processing, count the BI writes during a peak period to ensure that you allocate enough space to handle peak days. If the workload grows, make sure you increase the amount of space available for AI files. Always have extra disk space available for unanticipated growth of the AI file.

Another way to calculate your AI file size is to use **Before-Image Cluster Size** from the PROMON utility **BI Log Activity Display** and the **Checkpoints** from the **Activity Display.** If you want your AI file to hold all the changes made over a two hour period, determine how many checkpoints occurred during a two hour time period of peak activity. The number of checkpoints multiplied by the BI cluster size equals the minimum size your should make your AI extents to capture the changes. For example, if you have four checkpoints in two hours, and each cluster is 0.5MB, then you need your AI files to be at least 2.0MB to record two hours of activity.

Once you have sized your after-image areas, you need to manage them so that there is always space available to write in an extent. See Managing after-imaging files on page 144 for details on managing your after-image areas.

Creating after-image areas

You create after-image areas using PROSTRCT and a structure description file. Use PROSTRCT CREATE to create after-image areas when you first define your database, and PROSTRCT ADD to add after-image areas to an existing database.

To create AI areas when defining your database:

1. Create the structure description file for the database. See Creating and Deleting Databases on page 41 for a complete description of how to create a structure description file.
   a) Define the data extents.
   b) Define any BI extents.
   c) Define any fixed-length AI extents. You must define four tokens in the file description line.

      The table below describes each token and value you enter for fixed-length files.

Table 27: File tokens

Token	Description
Storage area type	Specifies the file type. Type `a` to indicate that this extent is an AI extent.
Extent path and file name	Specifies the extent file pathname. Enter a pathname that represents a standard operating system file. If you do not supply an extension, the database engine automatically appends a `.a` extension.
Extent type	Specifies whether the extent is a fixed-length or variable-length extent. Type `f` for a fixed-length extent. It must be lowercase.
Extent size	Specifies the length of the extent in 1,024-byte (1K) units. The minimum length for a fixed-length extent is 16K. The extent length must be a multiple of 16K for all operating systems. If you specify any other size, PROSTRCT displays a warning message and rounds the size up to the next multiple of 16K. The maximum length of an extent depends on the size of the file system and/or physical volume that contains the extent.

The following example shows a fixed-length after-image extent definition:

```
a    db/mary/apdir/test.a1 f 2048
```

d) Define any variable-length after-image extents. Only specify the first two tokens: the extent type and pathname. Unlike data or BI extents, you can define more than one variable-length after-image extent for a database.

2. Create the empty database structure using the PROSTRCT utility. For more information about the structure description file and PROSTRCT, see Maintaining Database Structure on page 325.
3. Create the initial database contents using the PROCOPY or PROREST utility. For more information about the PROCOPY and PROREST utilities, see Other Database Administration Utilities on page 821

Adding AI areas to an existing database

Use the following procedure to add AI areas to an existing database:

1. Create a structure definition file containing the descriptions of the files you are adding to your database. See Table 27: File tokens on page 142 for the tokens used to describe the files.
2. Use PROSTRCT ADD to add the new files to your database. For more information about the structure description file and PROSTRCT, see Maintaining Database Structure on page 325.
3. Use PROSTRCT LIST to save your updated database definitions in a structure file.

Enabling after-imaging offline

Use the RFUTIL utility with the AIMAGE BEGIN qualifier to enable after-imaging. The use of after-imaging requires a recent backup of the database because the database backup and AI files are treated as a recoverable set of files.

To enable after-imaging for an offline database:

1. Create the AI areas in your database. See Creating after-image areas on page 141 for detailed instructions.
2. Back up your database.
3. After you back up your database, enable after imaging.

   Use the following command to enable after-imaging:

```
rfutil db-name -C aimage begin
```

After imaging is enabled. However if the database has been modified since it was last backed up, RFUTIL displays an error message and does not enable after-imaging.

Enabling encrypted after-imaging offline for an encryption-enabled database

Encrypting after-image files for an encryption-enabled database follows security best practices. By default, enabling your database for encryption, also enables your AI files for encryption. However, you can explicitly enable encryption of your AI files when you begin after-imaging offline, as shown:

```
rfutil db-name -C aimage begin -aiencryption enable
```

Enabling unencrypted after-imaging offline for an encryption-enabled database

By default, enabling your database for encryption, also enables your AI files for encryption. However, you can explicitly disable encryption of your AI files when you begin after-imaging offline, as shown:

```
rfutil db-name -C aimage begin -aiencryption disable
```

Caution: Unencrypted AI files can contain your encrypted data in an unencrypted format. Not encrypting your AI files is a security risk.

Enabling after-imaging online

Use the PROBKUP utility with the `online` and `enableai` qualifiers to enable after-imaging on an online database. The completed backup is the baseline for rolling forward after-image extents.

To enable after-image for an online database:

1. Create the AI areas in your database. See Creating after-image areas on page 141 for detailed instructions.
2. Simultaneously backup your database online and enable after-imaging with PROBKUP.

   Use the command shown:

```
probkup online dbname output-device enableai
```

For complete PROBKUP syntax information, see PROBKUP utility on page 832.

Enabling encrypted after-imaging online for an encryption-enabled database

Encrypting after-image files for an encryption-enabled database follows security best practices. By default, enabling your database for encryption, also enables your AI files for encryption. However, you can explicitly enable encryption of your AI files when you begin after-imaging online, as shown:

```
probkup online dbname output-device enableai -aiencryption enable
```

Enabling unencrypted after-imaging online for an encryption-enabled database

By default, enabling your database for encryption, also enables your AI files for encryption. However, you can explicitly disable encryption of your AI files when you begin after-imaging online, as shown:

```
probkup online dbname output-device enableai -aiencryption disable
```

Caution: Unencrypted AI files can contain your encrypted data in an unencrypted format. Not encrypting your AI files is a security risk.

Managing after-imaging files

Managing after-imaging areas is a multi-step process. The manual steps involved include:

- Monitoring AI file status on page 144
- Switching to a new AI file on page 146
- Archiving an AI file on page 147
- Making an AI file available for reuse on page 149

Many of these steps can be automated using the AI File Management utility, however it is important to understand the manual steps involved in the process. For details on the utility, see AI File Management utility on page 150. The following sections detail each task in the manual process.

Monitoring AI file status

You must monitor the status of the AI extents. The AI extent status can be one of the following:

- **Empty** — When an AI extent is empty, it is available for use.
- **Busy** — When an AI extent is busy, it is currently in use.
- **Full** — When a busy AI extent fills up, an extent switch occurs: the state of the AI extent changes to full and the next AI extent becomes busy.

- **Locked** — When running OpenEdge Replication, a full extent is locked until the contents of the extent have been replicated to the target database. When the data is replicated, the extent is unlocked and marked Full.
- **Archived** — When running OpenEdge Replication, and AI File Management, an extent that was locked, and has been archived by the AI File Management utility, is marked as archived until OpenEdge Replication finishes replicating the extent. OpenEdge Replication will change the status of an archived extent to empty when the replication is finished.

There are three ways to determine the status of your AI files:

- **RFUTIL AIMAGE EXTENT LIST**

  Use RFUTIL AIMAGE EXTENT LIST to display information about each extent, including status:

```
rfutil db-name -C aimage extent list
```

  This command returns no information if after-imaging is disabled. For more information, see RFUTIL AIMAGE EXTENT LIST qualifier on page 808

- **PROMON**

  Use **PROMON** > **R&D** > **Status Displays** > **AI Extents** to display information about each extent, including status. For example:

```
09/13/05        Status: AI Extents
15:28:44

   Area  Status     Type      File       Size     Extent Name
                              Number    (KBytes)

     13  BUSY       Fix          1          505   /usr1/101A/docsample.a1
     14  EMPTY      Fix          0          505   /usr1/101A/docsample.a2
     15  EMPTY      Var          0          121   /usr1/101A/docsample.a3

Enter <return>, R, P, T, or X (? for help):
```

- **RFUTIL AIMAGE QUERY**

  The AIMAGE QUERY qualifier to RFUTIL provides you with a tool to query the status of a specified AI extent. The output of AIMAGE QUERY is intentionally brief. For most queries, the result returned will be a single number or word. The output is designed to be easily incorporated into scripts for managing your after-imaging.

  Use the following command to query an AI extent:

```
rfutil db-name -C aimage query query-option by search-option search-value
```

  The *query-option* specifies the information you want to gather about the AI extent. The *search-option* specifies the how you are identifying the AI extent to query. The *search-value* specifies the match criteria for the *search-option*. For example, if you want to know when after-imaging started writing to a particular extent, identified by its sequence number, use the following command:

```
rfutil db1 -C aimage query startdate by sequence 1
```

The output returned is the date:

```
Wed May 26 15:06:49 2004
```

If you want all the information about an extent, specified by its name, use the following command:

```
rfutil db1 -C aimage query all by name db1.a1
```

The output returned is:

```
Extent:  1
Status:  Busy
  Type:  Fixed Length
  Path:  /dbshare/databases/db1.a1
  Size:  120
  Used:  1
 Start:  Wed May 26 15:06:49 2004
 Seqno:  1
```

For more complete syntax detains, including the *query-option* and *search-option* values, see RFUTIL AIMAGE QUERY qualifier on page 810.

RFUTIL AIMAGE EXTENT FULL displays the filename of the oldest full file. You can then use this information to archive extents in the order in which they were filled. Although there might be multiple full files, this command displays the pathname of the oldest full file:

```
rfutil db-name -C aimage extent full
```

For more information, see the RFUTIL AIMAGE EXTENT FULL qualifier on page 1381.

Switching to a new AI file

You switch to a new AI extent for the following reasons:

- As part of the backup schedule
- When the current fixed-length AI extent is full, or when the disk holding the current variable-length AI extent is full
- Before archiving an AI extent

Except when you switch to a new extent because the current extent is full, switching to a new AI extent establishes a starting point for backup; after you restore the backup, you roll forward starting from that extent.

Note: When you perform an online backup, PROBKUP automatically switches over to a new extent as long as the next extent is empty. Before you perform the online backup, make sure that the next extent is empty.

A fixed-length extent has a predefined size, so the database engine can determine when the extent becomes full.

In contrast to a fixed-length extent, a variable-length extent does not have a predefined maximum size. Therefore, the database engine cannot anticipate when the extent is about to become full. Unless you force a switch using RFUTIL AIMAGE NEW, the database engine continues writing to the extent until an operating system limitation is reached, you reach the 2GB addressable AI file limit without large files enabled, or there is no more room left on the disk. When the extent becomes full, the database engine automatically switches to the next extent, provided that the next extent is empty. For more information on large files, see Database creation and large file support on page 42.

If the next extent is full, the database engine shuts down the database. However, you can use the After-image Stall (`-aistall`) parameter to suspend database activity and send a message to the log file or you can use the RFUTIL qualifier AIMAGE AIOFF to disable after-imaging. If you use `-aistall`, you can archive the oldest full extent and mark it as empty. The system will then automatically switch to that extent and resumes database activity automatically. For more information on the `-aistall` parameter, see Database Startup Parameters on page 473. If you use RFUTIL AIMAGE AIOFF, after-imaging becomes disabled and can no longer write notes.

Note: You can only use the `-aistall` parameter and RFUTIL AIMAGE AIOFF in multi-user mode.

When the database engine suspends database activity or shuts down the database, it sends the following message to the log file:

```
Can't switch to after-image extent filename it is full.
Backup ai extent and mark it as empty (3774)
```

The database engine cannot resume database activity until the next extent is backed up and marked as empty.

Switch to the next extent in the sequence

You can manually perform an online AI extent switch if you want to archive the AI file at regularly scheduled times instead of waiting until the extent becomes full.

To switch to the next extent in the sequence:

1. Make sure the next extent in the sequence is archived, and if it is not, then archive it. See Archiving an AI file on page 147 for details.
2. Use the RFUTIL AIMAGE NEW utility.

   The syntax for RFUTIL AIMAGE NEW is as follows:

   ```
   rfutil db-name -C aimage new
   ```

When you issue the RFUTIL AIMAGE NEW command, RFUTIL changes the status of the current extent to full and changes the status of the next file to busy. For more information on this command, see RFUTIL Utility on page 797.

Archiving an AI file

Backing up the AI file involves:

- Scheduling backups on page 148
- Performing the backup on page 148

- Protecting the backup on page 148

Scheduling backups

Depending on your database backup schedule and needs, you might want to schedule AI file backups:

- Each time you perform a database backup

  Back up the AI files each time you back up the database. It should be a routine part of every backup. Also, back up the AI files on a backup media different from the database, log, and BI files. This technique provides a way to reconstruct the database by rolling forward the backup of the AI files against the previous backup of the database.

  If you do choose to back up the AI files on the same media as the database, it is essential that you back up the AI files in a way that lets you restore them separately from the database, BI, and log files.

  Note: If you back up the AI files when you perform a database backup, keep in mind that the AI files backed up are used with the previous database backup.

  You can use the RFUTIL utility commands with online databases that have AI extents. You can perform online backups while after-imaging is enabled. The backup utility automatically marks the busy AI file as full and switches over to a new AI file.

- On a daily basis

  You should consider backing up the AI files on a daily basis if:

  - You are on a weekly full backup schedule and the AI files will grow very large during the week.
  - You want to perform daily backups of the AI files instead of performing incremental backups. However, it is quicker to restore your database from incremental backups than by rolling forward AI files.

If you are using a single AI file, it is important to back up the AI file before you fill the disk that contains it. If you do not back it up in time, the database engine shuts down the database. For a complete description of how to recover from a full AI disk, see Recovering a Database on page 117. Also, if you are using a single AI file, you must shut down the database to switch to a new AI file.

If you are using multiple AI extents, you must back up the extents regularly to ensure that the system does not run out of AI space. Before deciding to back up the AI files every day, consider that recovering the database from small AI files is more intricate than recovering from a single, large AI file.

Performing the backup

You must use an operating system utility to back up the AI files regardless of whether you are using a single AI file or multiple AI files. Ensure that the backup technique backs up the entire file. On many UNIX systems, certain utilities (for example, `cpio`) will back up only the first part of files over a certain size (controlled by the `ULIMIT` parameter). Backups of partial AI files are invalid and unusable. If you use `ftp` to transfer the AI files to a different machine, binary mode must be used. Failing to use `ftp` in binary will leave your AI files in an unusable state.

Protecting the backup

After you back up the AI file, make sure you:

- Label the backup. Properly labeling backups helps you ensure database integrity. Include the following on the label:
  - Date and time of the backup

- Name of the database
- Name and type of the file
- Volume number and total number of volumes of the media, even if there is only one volume
- Initials of the person who made the backup
- Utility to use to restore the backup

- Keep all backups for at least two weeks.

  In the event that you have to roll the database forward, you can reapply each of the backups to recover the lost data. If you do not keep a backup of the AI files and you lose the database and AI files, the most recent backup might be as much as a week old. However, if you have been doing daily backups of the AI files, you can use the most recent copy of that file to reconstruct a more recent version of the database.

- Keep backups in an area other than where the computer is located, preferably in another building.

  In the event of building damage, you are less likely to lose both the online and backup versions of the files.

Extracting AI blocks

Extracting AI blocks from an AI extent compacts your AI data into the smallest possible file. Extracting AI blocks can significantly shrink the size of an AI file, particularly if you have large fixed-size AI extents which are not filled to capacity. Extracting AI blocks is beneficial in cases where you are using after-imaging to replicate data, perhaps at a disaster recovery site, and you need to minimize file transfer size.

Use the AIMAGE EXTRACT qualifier of the RFUTIL utility to extract AI blocks from an extent, as shown:

```
rfutil db-name -C aimage extract -a ai-extent -o output-file
```

The output file of extracted blocks is equivalent to the source AI extent. Use the file of extracted blocks with the RFUTIL utilities to roll forward a target database. For complete syntax information see RFUTIL AIMAGE EXTRACT qualifier on page 809.

Note: Extracting blocks from an AI extent is only beneficial for fixed length extents that are not filled to capacity. There will be minimal savings of disk space when extracting blocks from a variable length extent.

Making an AI file available for reuse

After you back up an AI file, you must mark the file as empty so that the database engine can reuse the file. To mark a file as empty, use the AIMAGE EXTENT EMPTY qualifier of the RFUTIL utility. After you mark the file as empty, the database engine overwrites the contents of the file.

Use the following command to mark the AI file as empty:

```
rfutil db-name -C aimage extent empty [extent-number | extent-path]
```

If you do not specify an *extent-number* or *extent-path*, RFUTIL automatically marks the oldest full extent as empty.

Note: Use the RFUTIL AIMAGE EXTENT LIST or RFUTIL AIMAGE EXTENT FULL utility to determine the *extent-number* or *extent-path*.

AI File Management utility

The AI File Management utility provides customers with a best-practice approach to managing after-image extents. The utility has three major goals:

- Archive FULL AI extents to a user-specified location
- Maintain a log file to aid in ROLL FORWARD
- Be tightly integrated with OpenEdge Replication

AI File Management has two modes: automatic and manual. The automatic mode allows users with little or no experience with after imaging to quickly get started. In automatic mode, the utility handles AI extent archival for you. In the manual mode, the knowledgeable user has greater control over the process of archiving AI extents.

Automatic extent archiving

Automatic extent archiving is performed by a daemon started by the database broker. The daemon operates in two modes: timed and on-demand. Operating in timed mode, an AI extent switch is performed at the specified interval, in addition to an implicit switch when an extent fills. The minimum time interval is 120 seconds, and the maximum interval is 86400 seconds (24 hours.) Operating in on-demand mode, extents are only switched when they become FULL. In either mode, FULL extents are archived every five seconds. When operating the AI File Management utility daemon, you must have sufficient AI extents to keep at least one extent available for writing at all times.

Timed mode archiving process without OpenEdge Replication

When the time interval expires for the AI File Management utility daemon, the following process executes to archive extents:

1. Awake from five second sleep and archive all FULL AI extents.
2. Check to see if the time interval has expired. If the interval has expired, switch the current extent. Switching causes the current BUSY extent to be marked FULL, and the next EMPTY extent to be marked BUSY.
3. Mark archived extents as EMPTY.
4. Sleep for five seconds.

It is possible that there will be no FULL extents to archive on many iterations of this loop. After the timer expires, there will be least one FULL extent to archive, the one marked FULL in Step 2. On a busy system, it is possible that additional extents fill during Step 3 and Step 4 of the archiving process. They are archived the next time the daemon awakes.

Timed mode archiving process with OpenEdge Replication

When you have OpenEdge Replication enabled on your database, the daemon follows a similar process for archiving extents:

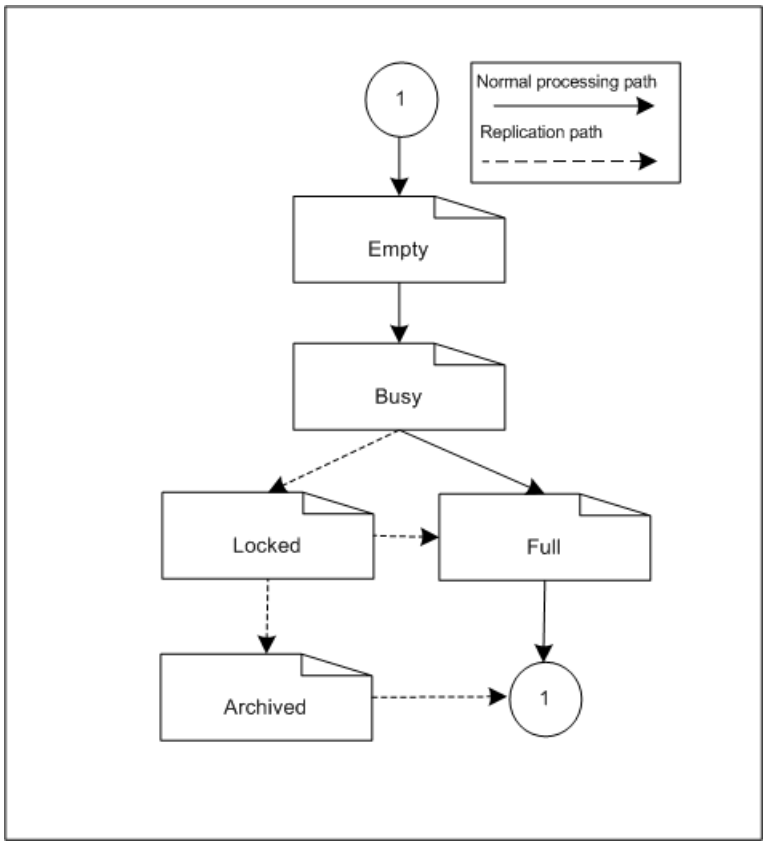
1. Awake from five second sleep and archive all FULL and LOCKED AI extents.
2. Check to see if the time interval has expired. If the interval has expired, switch the current extent. Switching causes the current BUSY extent to be marked LOCKED, and the next EMPTY extent to be marked BUSY.

3. Mark archived FULL extents as EMPTY and LOCKED extents as ARCHIVED.
4. Sleep for five seconds.

The difference in the archiving process when OpenEdge Replication is enabled is that extents cannot be emptied until they have been fully replicated. Extents transition from the BUSY state to the LOCKED state. If a LOCKED extent is replicated before it is archived, it transitions to the FULL state and the AI File Management daemon archives it. If a LOCKED extent is archived before it is replicated, it transitions to an ARCHIVED state, and it becomes the responsibility of OpenEdge Replication to transition it to an EMPTY state when replicated.

The following figure shows the state transitions graphically.

Figure 13: AI extent state transitions



On-demand archiving process without OpenEdge Replication

When the AI File Management daemon is operating in on-demand mode, the following process executes to archive extents:

1. Awake from five second sleep. Archive all FULL AI extents, and mark as EMPTY.
2. Normal database activity causes the current BUSY extent to fill. The extent is marked FULL, and the next EMPTY extent is marked as BUSY.
3. Sleep for five seconds.

On-demand archiving process with OpenEdge Replication

When the AI File Management daemon is operating in on-demand mode and OpenEdge Replication is enabled on your database, the following process executes to archive extents:

1. Awake from five second sleep. Archive all FULL and LOCKED AI extents. LOCKED extents are marked as ARCHIVED; FULL extents are marked EMPTY.
2. When OpenEdge Replication has replicated the LOCKED extent, it will mark the extent EMPTY.
3.
4. Normal database activity causes the current BUSY extent to fill. The extent is marked LOCKED, and the next EMPTY extent is marked as BUSY.
5. Sleep for five seconds.

Enabling your database for automated AI file management

If you have enabled after imaging for your database, the following examples demonstrate the steps to automate your AI file management. If you do not have after imaging enabled, you can enable after-imaging and AI file management simultaneously when your database is online. Otherwise see Enabling after-imaging offline on page 142. The example uses a database named `mydb`.

Enabling an offline database for automated AI file management

Use the following procedure to enable an offline database for automated AI file management:

1. Verify that your database has after-imaging areas and after-imaging is enabled. If not enabled, see Enabling after-imaging offline on page 142 for instructions.
2. Create directories on disks with sufficient space to archive your AI extents.
3. Enable your database for AI file management.

   Use the following syntax, where your database is named `mydb`:

```
rfutil mydb -C aiarchiver enable
```

4. Start your database broker with AI file management parameters:

   Use the following syntax, where your database is named `mydb`:

```
proserve mydb -S 6931
         -aiarcdir /usr1/aiarchives/mydb/,/usr2/aiarchives/mydb/
         -aiarcinterval 120
```

Enabling an online database for automated AI file management

Use the following procedure to enable an online database for automated AI file management:

1. Verify that your database has after-imaging areas and after-imaging is enabled. If not enabled, see Enabling after-imaging online on page 143 for instructions.
2. Create directories on disks with sufficient space to archive your AI extents.
3. Enable your database for AI file management with an online backup. The backup provides the baseline for rolling forward extents.

Use the following syntax, where your database is named mydb:

```
probkup online mydb /usr4/backups/mydb.bck enableaiarchiver -aiarcdir
           /usr1/aiarchives/mydb/,/usr2/aiarchives/mydb/
         -aiarcinterval 120
```

Additional info for enabling automated AI file management

In both examples, two directories are specified as output destinations for the archived AI files, /usr1/aiarchives/mydb and /usr2/aiarchives/mydb. If you do not create the directories, you can add an additional parameter, -aiarcdircreate, to instruct the AI file management utility to create the archive directories. In Windows, if you specify more than one directory, place the directory list in quotes. AI File management switches to the second directory when there is insufficient space to write to the current directory (the disk is full). If you specify multiple archive directories, they should be on separate partitions or disks. The -aistall parameter is implicitly included.

Setting -aiarcinterval to 120 directs the daemon to waken and archive extents every 2 minutes. You can specify an interval as small as 2 minutes, and as large as 24 hours. If you omit the -aiarcinterval parameter, the daemon will waken each time an archive fills, regardless of the amount of elapsed time.

If you are unsure of a time interval, you can initially omit -aiarcinterval, and then switch to a timed interval based on the frequency of archives as recorded in the archive log file.

Monitoring and adjusting automated AI File Management

Once you have started automated archiving of your AI extents, you need to monitor the performance of the daemon and the capacity of your archive directories. RFUTIL provides you with qualifiers to modify your archive process online. The following commands continue with the example started in the previous section.

- To modify the directory list where your AI extents are archived, use the following command:

  ```
  rfutil mydb -C aiarchiver setdir
               /usr3/aiarchives/mydb/,/usr4/aiarchives/mydb/
  ```

  The directories specified with this command replace any previously specified archive directories. If the directories do not exist, you can add an additional parameter, -aiarcdircreate, to instruct the AI file management utility to create the archive directories.

- To modify the time interval for your AI file management utility daemon, use the following command:

  ```
  rfutil mydb -C aiarchiver setinterval 600
  ```

  This command increases the timer of the file management daemon to five minutes.

- To force the next FULL AI extent to be archived immediately, use the following command:

  ```
  rfutil mydb -C aiarchive nextextent
  ```

This command archives the next FULL extent to the previously specified directory list. If the daemon is operating in timed mode, the timer is reset. If your database stalls because all your AI extents are full, this command will free an extent. After clearing the stall, set the archive interval to a smaller number or to 0 prevent another stall.

For complete syntax of the AIARCHIVE and AIARCHIVER qualifiers to RFUTIL, see RFUTIL Utility on page 797

Restarting the AI file management utility daemon

If the AI file management utility daemon terminates abnormally or is stopped using AIARCHIVER END, it can be restarted without restarting the database server.

To restart the AI file management utility daemon, use the following command:

```
PROSHUT dbname -C aimgt -aiarcdir archive-directory-Path -aiarcinterval n
[-aiarcdircreate]
```

PROSHUT will not shutdown the database server when the `-C` option is specified.

Archived extents

Extents archived automatically by the AI File Management utility daemon are assigned an unique file name by the archive process. The archived file name is comprised of a combination of file names and sequence numbers that identify the source of the archive file and the state of the source database at the time of the archive. The elements are:

- The full file specification of the source database, with a "~" replacing the directory separator
- In Windows, the colon (:) following a drive letter is replaced by an exclamation point (!)
- The AI Sequence number or file number
- The name of the AI extent

The elements are concatenated into a file name as shown:

```
directory~spec~database.ai-sequence.extent-name
```

For example, if the name of your archive file is `usr1~dbs~sales.03.sales.a3`, you can reconstruct the following information:

- The source database is `/usr1/dbs/sales.db`
- The AI sequence number is `03`
- The extent name is `sales.a3`

Archive log file

The AI File Management utility creates a log based on various activities that occur while the database is enabled for AI File Management. The archival log file is written to the same directory as your database `.db` file, and is named, *`database-name`*`.archival.log`. The log file is created the first time information is written to it. The information stored in the archive log file provides details of when the AI file management utility was started, database backups, and extent archiving activity that you can use to develop either an automated or manual recovery of a database. This file is maintained by the OpenEdge RDBMS; it can be read, but must not be altered.

The archival log is a text file, organized into logical records that are separated by platform dependent end-of-line characters. The records have the following characteristics:

- Each line in the file is a record.
- Records that begin with a "#" are comments, and are purely informational.
- Records contains multiple fields that are separated by commas.
- The first field of each record is the record type.
- The organization of a records may change from one OpenEdge release to another.
- New records types may be added in future OpenEdge releases.
- The Version Indicator Record (record 0255) indicates the version of the records that follow. The version number in the Version Indicator Record is updated if there is a change in the format of any existing archival log record or a new archival log record type is added,

The table below describes the record types supported in the archival log file.

Table 28: Archival log record types

Record type value	Description
0255	**Version indicator** — This record indicates the version of archival log records that follow.
0001	**Automatic after-image file archive** — This record describes the archival of an after-image extent by the AI File Management utility.
0003	**Manual after-image file archive** — This record describes the manual archival of an after image file by the RFUTIL AIARCHIVE EXTENT command.
0004	**Manual after-image file archive to the NULL device** — This record describes the manual archival of an after image file by the RFUTIL AIARCHIVE EXTENT command to the NULL device.
0032	**Full backup** — This record describes a FULL online or offline backup.
0033	**Incremental backup** — This record describes an INCREMENTAL backup.

Each time the AI File Management daemon is started, a header line is written to the archive to record the date, time, using the following format:

```
# After-image Extent Manager started on : Tuesday October 11, 2005 at 10:02:25
```

Version indicator record

The version indicator record is written to the archival log file anytime a process initially writes to the file. The format or layout of this record will change from release to release. This record describes the format of each archival log record that follows. The version indicator record provides version and startup information and is formatted as follows:

```
0255, date, time,version
```

The table below describes the fields in a version indicator record.

Table 29: Format of an archival log startup line

Field	Value
Record type (0255)	Header line indicator.
date	Date the AI File Management utility started in YYYYMMDD format.
time	Time the AI File Management utility started in HHMMSS format.
version	Version of the archival log records that follow. This field facilitates the programmatic parsing of the archive log file.

Archive record

Each time an after-image extent is archived, an archive record is written to the archival log. The archival record describes the archival itself and the after-image file being archived An archive line is formatted as follows:

```
Record type,database,date,time,aibegin-date,aibegin-time,aiseq,
        extent-name,target-extent-name
```

The table below describes each field of an AI extent archive record.

Table 30: Format of an AI extent archive record in the archive log file

Field	Value	Current field position	Prior field position[2]
Record type	AI extent archive record values are: • 0001 — Automated extent archival • 0003 — Manual extent archival • 0004 — Manual extent archival to a NULL device	1	1
database	Name of the database	2	2
date	Date of the backup in YYYYMMDD format	3	3
time	Time of the backup in HHMMSS format	4	4
aibegin-date	Date that after imaging was started for the database in YYYYMMDD format	5	5
aibegin-time	Date that after imaging was started for the database in HHMMSS format	6	6
buseq	Backup sequence number of the backup	Removed	7
aiseq	AI extent sequence number, with leading zeros (number of AI extents since last AIMAGE BEGIN)	7	8
extent-name	Source name of the archived extent	8	9
target-directory	The directory where the extent is archived	Removed	10
target-extent-name	The file name of archived extent	9	11

Backup record

A backup record is formatted as follows:

```
Record type,database,date,time,online indicator,buseq,backup-date,
      backup-time,[incr-seq,]aiseq,backupset-name,target-count,
      backup-targets
```

The table below describes each field of a backup record.

[2] **Field record position for an AI extent archive record prior to OpenEdge Release 10.1C02.**

Table 31: Format of a backup record in the archive log file

Field	Value	Current field position	Prior field position[3]
Record type	Backup indicator record values are: • 0032 — Full backup • 0033 — Incremental backup	1	1
database	Name of the database	2	2
date	Date the backup record was written to the archive log file in YYYYMMDD format	3	3
time	Time the backup record was written to the archive log file in HHMMSS format	4	4
*online indicato*r	Online indicator; possible values are: • 0 — offline • 1 — online	5	5
buseq	Backup sequence number of the backup	6	6
backup-date	Date that the backup was started for the database in YYYYMMDD format	7	7
backup-time	Date that the backup was started for the database in HHMMSS format	8	8
incr-seq	For incremental backups only, the incremental backup sequence number; if a full backup, this field is omitted	9 (for 0033)	9 (for 0033)
aiseq	AI extent sequence number, with leading zeros (number of AI extents since last AIMAGE BEGIN)	9 (for 0032) 10 (for 0033)	9 (for 0032) 10 (for 0033)
backupset-name	The backup set name—the format is `BACKUP_SET_`*`YYYYMMDD_HHMMSS`* where the date and time components reflect the `backup-date` and `backup-time` values	10 (for 0032) 11 (for 0033)	12 (for 0032) 13 (for 0033)
target-count	Number of backup target files	11 (for 0032) 12 (for 0033)	10 (for 0032) 11 (for 0033)
backup-targets	Comma separated list of backup target names	12 (for 0032) 13 (for 0033)	11 (for 0032) 12 (for 0033)

[3] **Field record position for a backup record prior to OpenEdge Release 10.1C02.**

Database recovery with an archival log

When after-image file management is enabled, a record of each backup and each after-image file archived is written to the archival log. This information can be used to aid in the recovery of a damaged database provided there are complete sets of backup and AI extent archive records in the archival log file. For recovery, a complete set of archival records consists of the following:

- A full backup record
- One or more incremental backup records
- One or more after-image file archive records

The information from all three records must be matched together as follows:

1. Start with a FULL backup.
2. Match INCREMENTAL backups to a FULL using the backup sequence number (field 6 from the 0032 records) and the backup sequence number (field number 6 in the 0033 records). If there are INCREMENTAL backups with the same backup sequence number as the FULL backup record, they must be sequenced by the incremental backup number (field 7 from the 0033 record).
3. Match the AI archives to the backups using the AI sequence number (field 9 from the 0032 record or field 10 from the 0033 record) to retrieve all after-image file archival records where the after-image file sequence number from the backup record is >= to field 7 of the after-image file archival records.

Once the correct records are identified, they can be used to drive the database recovery process using the backup volume names from the backup records and the target filename from the archival records.

Add and reorder AI extents

Additional AI extents are needed when you are unable to execute an extent switch because all extents are full, and you are unable to empty the next extent in the sequence for reuse. If an extent switch cannot be performed, your database shuts down or stalls if you have enabled after-image stall.

To add additional AI extents, create a `.st` file to define the additional extents, and use PROSTRCT ADD or PROSTRCT ADDONLINE to update your database. See Using the OpenEdge Structure Add utility on page 329 for details. When you add additional AI extents, they are added to the end of the extent list. PROSTRCT REORDER AI will move the new, empty AI extents so that they immediately follow the busy extent. Reordering your extents is necessary if the next extent in the switch sequence is not empty. This condition can occur if OpenEdge Replication has any of your full extents locked. The Replication lock blocks the user from using RFUTIL to empty the extent so that it may be reused. For more information on OpenEdge Replication, see *OpenEdge Replication: User's Guide*.

Use the following command to reorder your AI extents:

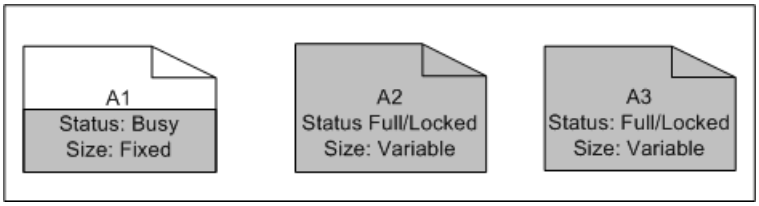
```
prostrct reorder ai db-name
```

Caution: PROSTRCT REORDER AI will rename AI extent files, changing the `.an` extension. Area numbers for the reordered extents will also change. It is critical that you use PROSTRCT LIST to generate an accurate `.st` file for the database after running this utility. For more information on generating a new `.st` file, see PROSTRCT LIST qualifier on page 792.

The following example illustrates the effect of PROSTRCT REORDER AI on your AI extents:

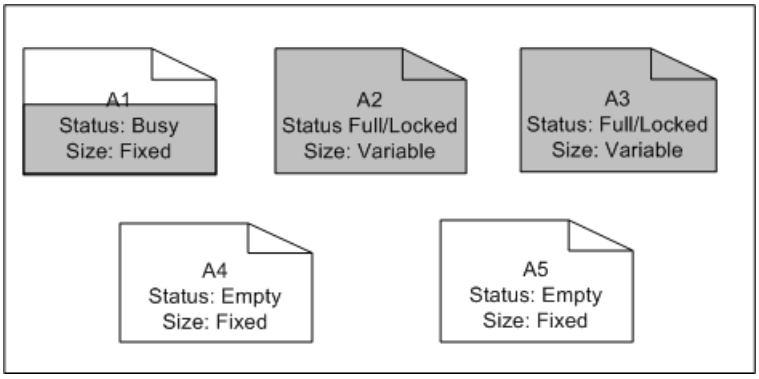
- The following figure shows the original AI extent configuration. The database will shutdown when `A1` fills because it cannot switch to the full `A2` extent.

  Figure 14: Original database AI extents



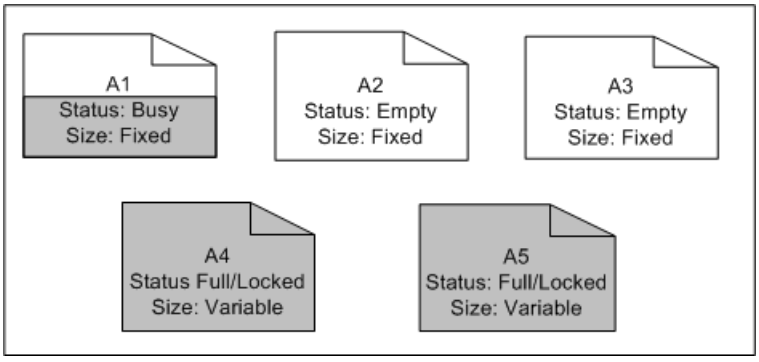
- The user must first add additional AI extents as the following figure shows.

  Figure 15: Database AI Extents after PROSTRCT ADD of two new extents



- Adding the new extents is not sufficient to prevent a database shutdown. The database will still shutdown when `A1` is filled because after-imaging still can not switch to the full `A2` extent. Use PROSTRCT REORDER AI to arrange the extents. The following figure shows the result of the PROSTRCT REORDER AI.

  Figure 16: Database AI extents after PROSTRCT REORDER AI



- After the reorder is complete, after-imaging can successfully switch extents. The result of the reordering is that the empty extents are moved to immediately follow the busy extent. Their area number and file name extensions have been altered to reflect the move. The physical location and size of the file on disk has not been altered.

Performing roll-forward recovery

To perform roll-forward recovery, roll forward all of the extents used since the last backup, in the order they were filled. You must roll forward all of the extents before you can start the database. If you inadvertently start a database before you restore all of the AI extents, the database engine rolls back any incomplete transactions. Although these transactions might be completed in other as yet unrestored AI files, they appear to the database engine as incomplete during the roll-forward process. If you start the database before you restore all of the extents, you must begin the roll forward again from the last backup.

Before performing roll-forward recovery, you must have the most recent database backup, all AI files generated since the most recent backup, and no missing files. The last file is usually the file in use when the system failed. If any AI file is missing, recovery can be performed up to the missing file, but no further.

Use the ROLL FORWARD qualifier of the RFUTIL utility to restore each file. The syntax is:

```
rfutil db-name -C roll forward -a ai-name
```

For complete syntax information see the RFUTIL ROLL FORWARD qualifier on page 1396.

The ROLL FORWARD qualifier fails if:

- You omit the After-image Filename (`-a`) parameter
- It cannot open the AI file
- You name the wrong AI file

If the system fails while you are running the ROLL FORWARD operation, restore the database files again and rerun the ROLL FORWARD operation.

The ROLL FORWARD qualifier always disables after-imaging for the database before beginning the roll-forward operation. After the roll-forward has completed, you must re-enable after-imaging with the AIMAGE BEGIN qualifier if you want continued AI protection.

To perform a partial roll-forward recovery, use the `endtime` or `endtrans` options. The `endtime` option lets you roll forward an AI file to a particular point. The `endtrans` option lets you roll forward an AI file to a particular transaction. For information on using these options, see RFUTIL ROLL FORWARD qualifier on page 815.

For more information about the ROLL FORWARD qualifier, see Recovering a Database on page 117 and RFUTIL Utility on page 797.

Roll forward with the OPLOCK qualifier

Typically, the roll forward process involves multiple AI files, with each AI file being processed by a separate roll forward session. This sequence of sessions should not be interrupted until the final AI file is processed to complete all the phases of the crash recovery.

The OPLOCK qualifier provides the ability to enforce a protection mechanism during the roll forward process, so that there can be no interruption to the target database before the final roll forward session is completed. This option must be enabled together with the rolling forward of an AI file. For example:

```
rfutil db-name -C roll forward oplock -a ai-name
```

Once the protection is enabled with OPLOCK, there are two ways to disable it:

- At the end of the final roll forward or roll forward retry session (after all the phases of crash recovery have been finished) the protection is automatically turned off.
- If you want to disable the protection (in situations where it is not desirable to roll forward to the last extent), use the OPUNLOCK qualifier as shown:

```
rfutil db-name -C roll opunlock
```

The OPUNLOCK qualifier:

- Disables the protection mechanism, making the database accessible to other users or utilities
- Completes the phases of the crash recovery

Note: The OPUNLOCK qualifier should only be used for situations with missing AI files. Once OPUNLOCK is run, the roll forward process stops.

Treatment of operations while under protection

Operations that do not go through the crash recovery phase are allowed when roll forward protection is enabled with the OPLOCK qualifier. These operations do not modify the target database, and are useful during the roll forward process. For example, you can use PROSTRCT ADD to add more data space so that the roll forward can continue without error.

Operations that go through the crash recovery phase are rejected under the protection because these operations cause the roll forward process to fail. One example is PROUTIL IDXBUILD. Operations that go through crash recovers are rejected even if the `-F` option is included on the command line, for example, PROUTIL TRUNCATE BI.

Because the OPLOCK qualifier restricts access during the roll forward, you may find that you cannot perform some operations that previously you could perform during the roll forward process. For example, attempting to modify the target database before the entire roll forward process is completed is not allowed while this option is invoked.

Messages that may appear on the screen and `.lg` file during the utility's operation include:

- Target database has been locked during roll forward process.
- Please wait until the roll forward process is finished or use the `rfutil -C roll opunlock` option to disable the protection.
- Target database has been unlocked during roll forward process.
- Roll forward process has been stopped with the opunlock option.

Rolling forward multiple files

To expedite the process of rolling forward multiple after-image extents, ROLL FORWARD accepts a file containing a list of after image files as input. The syntax of ROLL FORWARD with a list of extents is:

```
rfutil db-name -C roll forward -ailist listfile
```

The *`listfile`* file, containing a list of AI area files, must be structured as follows:

- Specify one AI file per line
- AI file names can be a fully specified filename or a filename relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

If a file in the list is not an after-image file, ROLL FORWARD issues a warning and the file is skipped.

Applying after-images files from a list

The following example demonstrates applying the after image files from a database named `source_db` to a database named `target_db`.

Use the following procedure example to apply after-image files from a list:

1. Create a list of after-image files from `source_db`.

   One method is as shown:

```
proenv>ls source_db.a* > source_ai_files.lst

proenv>cat source_ai_files.lst
source_db.a1
source_db.a10
source_db.a11
source_db.a12
source_db.a2
source_db.a3
source_db.a4
source_db.a5
source_db.a6
source_db.a7
source_db.a8
source_db.a9
```

   Notice that the files are sorted using an alphabetical sort. ROLL FORWARD sorts the files into the proper sequence order before applying them to the target database.

2. Create your target database by restoring a backup of your source database.

   The syntax is as shown:

```
proenv>prorest target_db source_bck.bkup
```

3. Apply the source after-image files to the target.

   The syntax is as shown:

```
proenv>rfutil target_db -C roll forward -ailist source_ai_files.lst
```

ROLL FORWARD sorts and applies the after-image files, reporting the progress of each file as it is applied to both the screen and the database log file. The following is a portion of the output:

```
OpenEdge Release 11.3ALPHA as of Wed Mar 13 19:01:40 EDT 2013
***Beginning processing of After-image extent C:\OE113WRK\source_db.a1
(17047)
After-image dates for this after-image file: (1633)
    Last AIMAGE BEGIN Mon Mar 18 18:12:06 2013 (1640)
    This is aimage file number 1 since the last AIMAGE BEGIN. (1642)
    This file was last opened for output on Mon Mar 18 18:12:11 2013. (1643)

10% of aimage file processed (964 notes applied)...
20% of aimage file processed (1924 notes applied)...
.
.
.
90% of aimage file processed (8424 notes applied)...
8932 notes were processed. (1634)
0 in-flight transactions. (3785)
1 transactions were started. (1635)
1 transactions were completed. (11138)
At the end of the .ai file, 0 transactions were still active. (1636)
***Finished processing of After-image extent C:\OE113WRK\source_db.a1
(17048)

***Beginning processing of After-image extent C:\OE113WRK\source_db.a2
(17047)
After-image dates for this after-image file: (1633)
    Last AIMAGE BEGIN Mon Mar 18 18:12:06 2013 (1640)
    Last AIMAGE NEW Mon Mar 18 18:12:58 2013 (1641)
    This is aimage file number 2 since the last AIMAGE BEGIN. (1642)
    This file was last opened for output on Mon Mar 18 18:12:58 2013. (1643)
.
.
.
***Finished processing of After-image extent C:\OE113WRK\source_db.a11 (17048)
***Beginning processing of After-image extent C:\OE113WRK\source_db.a12 (17047)

After-image dates for this after-image file: (1633)
    Last AIMAGE BEGIN Mon Mar 18 18:12:06 2013 (1640)
    Last AIMAGE NEW Mon Mar 18 18:25:26 2013 (1641)
    This is aimage file number 12 since the last AIMAGE BEGIN. (1642)
    This file was last opened for output on Mon Mar 18 18:25:26 2013. (1643)

108977 notes were processed. (1634)
1 in-flight transactions. (3785)
0 transactions were started. (1635)
1 transactions were completed. (11138)
At the end of the .ai file, 0 transactions were still active. (1636)
Roll forward of the after-image busy extent is complete. (3786)
All roll forward for database target_db is complete. (3787)
***Finished processing of After-image extent C:\OE113WRK\source_db.a12 (17048)
```

After-image sequences

The SEQUENCE qualifier to the RFUTIL utility provides database administrators the ability to update after-image sequence numbers and improves your ability to maintain a hot standby database recovery strategy.

RFUTIL SEQUENCE updates the sequence number of your hot standby target database to match the sequence number of the `AI` extent that was BUSY at the time the source database was copied to create the hot standby.

After executing RFUTIL SEQUENCE on your standby target database, you can keep the target in sync with the source by rolling forward `AI` extents, starting with the extent that was BUSY when the copy occurred.

See RFUTIL SEQUENCE qualifier on page 819 for the command syntax.

A hot standby is typically maintained by first making a copy of your source database, and then regularly updating the database by rolling forward the AI extents from the source to the target. The preferred methods of making a copy of your database use the OpenEdge utilities PROCOPY or PROBKUP. However, if you use your operating system commands to make a copy, the SEQUENCE qualifier to RFUTIL is available to update your target database to expect the correct after-image extent in the roll forward sequence. Correcting the sequence is only necessary if there has been an AI extent switch on your source database **after** the database was last marked as backed up and **before** the operating system copy was made.

Sequence not required

The following code sample demonstrates the steps for making a hot standby database using an operating system copy command. In this example, the target database does not need its sequence corrected:

```
# Update your source database by adding after-imaging extents
#
prostrct add production_db add.st

# truncate the database BI file and mark as backed up
#
proutil production_db -C truncate bi
rfutil production_db -C mark backedup

# Activate after-imaging
#
rfutil production_db -C aimage begin

# Make a copy of the database using operating system commands
#
cp production_db* /db_standby

# Mark the source database as backed up
#
rfutil production_db -C mark backedup

# Fix the target database file specifications and make new database st file.
#
prostrct repair /db_standby/production_db /db_standby/standby.st
prostrct list /db_standby/production_db /db_standby/production_db.st

# At this point allow updates to the source database
#
# Roll forward the first AI extent from production_db to standby copy
#
rfutil /db_standby/production_db -C roll forward -a production_db.a1

# Continue allowing updates and regularly roll forward the AI extents from
# the production_db to the standby copy.
```

Sequence required

The following code example demonstrates the steps for making a hot standby database using an operating system copy command, but in this example, updates to the database occur at an earlier point in the process. These updates require that the sequence of the target database be corrected prior to rolling forward an AI extent:

```
# Update your source database by adding after-imaging extents
#
```

```
prostrct add production_db add.st

# truncate the database BI file and mark as backed up
#
proutil production_db -C truncate bi
rfutil production_db -C mark backedup

# Activate after-imaging
#
rfutil production_db -C aimage begin

# At this point allow updates to the source database. This activity will
# fill the first AI extent, production_db.a1, and causing a switch to the
# second AI extent, production_db.a2.

# Make a copy of the database using operating system commands
#
cp production_db* /db_standby

# Mark the source database as backed up
#
rfutil production_db -C mark backedup

# Fix the target database file specifications and make new database st file.
#
prostrct repair /db_standby/production_db /db_standby/standby.st
prostrct list /db_standby/production_db /db_standby/production_db.st

# Correct the sequence number of target database.
#
rfutil /db_standby/production_db -C sequence

# Roll forward the second AI extent from production_db to standby copy.
# This is the extent which was BUSY at the time the copy of the database
# was made.
#
rfutil /db_standby/production_db -C roll forward -a production_db.a2

# Continue allowing updates and regularly roll forward the subsequent AI
# extents from the production_db to the standby copy.
```

The target database must have its sequence corrected prior to attempting to apply an AI extent with roll forward. Roll forward disables after imaging, and it is impossible to correct the sequence once after imaging has been disabled. If the roll forward on the target database fails, you must recopy from the source, use RFUTIL SEQUENCE to correct the sequence number, and then roll forward.

Disabling after-imaging

You use the RFUTIL utility to disable after-imaging.

To disable after-imaging:

1. Back up any full extents.

   This step is important because when you disable after-imaging on a database with AI extents, the database engine marks all of the AI extents as empty. This means that you cannot access any information that you did not back up for roll-forward recovery.

2. Back up the extent that is currently busy.
3. Disable after-imaging. Use the AIMAGE END qualifier with the RFUTIL utility.

The syntax is as follows:

```
rfutil db-name -C aimage end
```

Note:

You can also use the AIMAGE AIOFF qualifier with RFUTIL. Use AIMAGE AIOFF when you need to **temporarily** disable after-imaging, such as during scheduled maintenance.

Disabling encryption of AI files when disabling after-imaging

By default, disabling encryption does not change whether or not your AI files are encrypted in an encryption-enabled database.

If you want to cease encryption of AI files when you next re-enable after-imaging, you can explicitly disable encryption of your AI files when you disable after-imaging either permanently or temporarily:

- If you are permanently disabling encryption and disabling AI encryption, use AIMAGE END as shown:

  ```
  rfutil db-name -C aimage end -aiencryption disable
  ```

  AIMAGE END marks all AI files as EMPTY, so all information not backed up or applied to your target database with ROLL FORWARD is lost.

- If you are temporarily disabling encryption and disabling AI encryption, use AIMAGE AIOFF, as shown:

  ```
  rfutil db-name -C aimage aioff -aiencryption disable
  ```

  Encrypted AI files remain encrypted.

Caution: Unencrypted AI files can contain your encrypted data in an unencrypted format. Not encrypting your AI files is a security risk.

8

Maintaining Security

As an OpenEdge database administrator, you want to ensure that only authorized users connect to a database, as well as prevent unauthorized users from modifying or removing database files and objects.

For more information about the security features of OpenEdge, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*. For more information about establishing a user identity, see *OpenEdge Getting Started: Identity Management*.

Caution: Before changing your database's security implementation, make a backup of the original database. This way you can restore the original database if you cannot undo some of the security measures you set. For example, if you are the sole security administrator and you forget your user ID or password, you might be denied access to the database and the data contained in it. However, if you have a backup of the original database, you can restore it. To prevent such a mishap, designate at least two security administrators for each database.

For details, see the following topics:

- Establishing an OpenEdge user ID and password
- Establishing authentication for your OpenEdge database
- Connection security
- Running a user report
- Schema security
- Operating systems and database security

Establishing an OpenEdge user ID and password

Every OpenEdge database connection has a user identity specified by a user ID. All forms of OpenEdge application security use this user ID to determine the user's authorization to access data.

OpenEdge authenticates any user who connects to a database, whether from an ABL application, an SQL client, or a database utility. You can define the user IDs and passwords that OpenEdge can authenticate at connect time in two ways:

- Using the character mode Data Dictionary, the Data Administration tool, or the Database Administration Console of OpenEdge Management or OpenEdge Explorer, you can define user IDs and passwords in the `_User` table accounts. The `_User` table is a hidden table.
- You can define user IDs and passwords in the user accounts of the operating system where OpenEdge runs.

In OpenEdge, user IDs have two parts, a user name and domain name. The domain name is the name of an OpenEdge security domain that is defined and configured in the database. The configuration of a security domain identifies how OpenEdge (including SQL) or an ABL application can authenticate the user identity represented by the user ID and specifies the user account system where the user's identity is defined.

In the `_User` table accounts, user identities are defined with both a user and domain name. In Windows user accounts, user identities might also be defined with a domain that corresponds to an OpenEdge domain. UNIX user accounts are only defined with a user name and support only a single OpenEdge domain.

Note that whether or not the account system where a user identity is defined includes domains, when a user attempts to log onto a database connection, OpenEdge identifies an appropriate OpenEdge domain to authenticate the user's identity, depending on how the user specifies their user ID. For more information, see OpenEdge user IDs on page 170.

OpenEdge user IDs

The form of the user ID required for OpenEdge user authentication depends on the account system where a user's identity is defined and the configuration of OpenEdge security domains in connected OpenEdge databases. The form of the user ID used for various security functions in an OpenEdge database also depends on whether the database is a non-multi-tenant database or a multi-tenant database.

OpenEdge user IDs are defined as either non-qualified or fully qualified. A non-qualified user ID contains only a user name and no domain name. A fully qualified user ID contains both a user name and a domain name separated by the `'@'` character (also referred to as the *domain delimiter*). Whether OpenEdge requires a non-qualified or fully qualified user ID depends on the database configuration and the security function where it is used.

In general, the exact character set, length, and case sensitivity of a user ID depend on the account system where the user ID is defined. The non-qualified user ID (user name) defined in the `_User` table accounts of an OpenEdge database can be a string of up to 32 characters. Like table names, these OpenEdge internal user IDs must begin with a character from `a-z` or from `A-Z`. The user name can consist of:

- Alphabetic characters
- Digits
- The following special characters: `# $ % & - _`

OpenEdge internal user IDs are **not** case sensitive: They can be uppercase, lowercase, or any combination of these. You can establish user IDs in a database through the various OpenEdge database administration tools (see Establishing an OpenEdge user ID and password on page 170.) You can also define a blank user ID, written as the empty string, `""`. OpenEdge also uses the blank user ID as a default user ID when none is provided for certain database connection situations.

An OpenEdge domain name can be a non-empty, case-insensitive string with maximum length of 64 characters from the following restricted character set:

- `a` to `z`
- `A` to `Z`
- `0` to `9`
- Any of the following seven special characters:

```
# $ % & - _ .
```

OpenEdge also defines a blank (`""`) domain name that you can configure in various ways. OpenEdge uses this blank domain when a user does not specify a domain with their user ID. For more information on using user IDs, user names, and domain names, see *OpenEdge Getting Started: Identity Management*.

Note: In the OpenEdge Data Dictionary and Data Administration tool, the fields to enter users in the `_User` table accounts are labeled assuming one form of user ID—the non-qualified user ID. The purpose of the **User ID** field is to enter this non-qualified value. You enter the domain name for a fully qualified user ID in a separate **Domain Name** field. There is also a **User Name** field displayed in these tools. However, this is not for the user name that you enter to define a non-qualified user ID, but is meant to hold an informative name, such as the first and last names of the individual who owns the specified user identity.

Caution: When you first create a database, all users (including the blank user) have security administrator privileges. Configure and test user authentication for the users whom you intend to identify as security administrators **before** you change the Security Administrator list; this will allow you to log in and not be locked out of the database. You should also define the security administrator roles **before** you change the default table and field permissions of the database. After you define the first non-blank user in the `_User` table accounts, the blank user no longer has security administrator privileges for all subsequent connections to the database. If you have not defined at least one non-blank user as a security administrator, you will not be able to modify the security functions of the database and can lock yourself out of the database as a result.

OpenEdge passwords

The format and content requirements of passwords, again, depends on the account system where a user's identity is defined. Passwords defined for users in the `_User` table accounts can be a string of up to 16 case-sensitive characters. (If you type more than 16 characters in the **Password** field, the additional characters are not accepted.) When you add the password to the `_User` table accounts, the password is encoded with the ENCODE function. Because OpenEdge internal passwords are case-sensitive, ENCODE returns different values for uppercase and lowercase input.

Validating an OpenEdge user ID and password

Depending on the application and OpenEdge startup command, when logging in, the user might be prompted for their user ID and password, or the might specify a user ID and password using the User ID (`-U`) and Password (`-P`) startup parameters.

If you establish valid user IDs in the `_User` table accounts, the OpenEdge RDBMS prompts for a user ID and password at connection time. Typically, the application does this by running the login procedure. The standard startup procedure, PROSTART, automatically runs a login procedure for each connected database. If the application uses another startup procedure, the developer should run a login procedure from that procedure.

The login procedure uses the SET-DB-CLIENT function to check the user ID and password entered by the user. The procedure initializes a client-principal object with these user credentials and passes this object to the function. The user has three attempts to enter the correct user ID and password for each database. If the user fails to do so after three attempts, SET-DB-CLIENT exits the user from the database. If the user ID and password combination is valid for the database, then SET-DB-CLIENT establishes that user ID for the connection.

If the application does not run the login procedure at connection time, or if the user bypasses the login procedure (by pressing **END-ERROR** when prompted for the user ID and password), the database connection's user ID retains the default assigned at connection time. You can prevent users from connecting to the database with the blank user ID and accessing data by establishing compile-time and run-time security.

Caution: If a user connects to a database with the default blank user identity, the user potentially has full access to the database. You can prevent users with a blank user ID from gaining unauthorized access to an OpenEdge database in two ways: 1) setting a database option to **Disallow Blank UserId Connections**, in which case the User ID (`-U`) parameter and a non-blank user ID are required to connect to the database, or 2) setting the security option to **Disallow Blank Userid Access**. The first option prevents blank users from connecting to the database in certain, but not all, scenarios; the second option prevents blank users from accessing any data in the database by explicitly revoking blank user ID access to all data in the database, and provides the most comprehensive protection from unauthorized access by a blank user. Also, remember to assign non-blank user IDs to the security administrator role **before** restricted access for blank users.

For more information about compile- and run-time security, see *OpenEdge Getting Started: Identity Management*, *OpenEdge Development: Programming Interfaces*, and *OpenEdge Deployment: Managing ABL Applications*.

Establishing authentication for your OpenEdge database

To prevent unauthorized users from accessing, modifying, or removing database files, you can establish appropriate database administration and user-access privileges. The process for establishing authentication depends on whether your database has tables used from the ABL (Advanced Business Language) only, tables used from SQL only, or tables used from both ABL and SQL.

Caution: If you plan to use OpenEdge SQL, as **your first task** be sure to create an SQL DBA account as the **first** user account. Grant the account DBA rights by executing the GRANT statement. For more information, see *OpenEdge Getting Started: Identity Management*.

Tables used from ABL only

By default, no user ID or password is required. You can assign a security administrator role. Among other security functions, a security administrator is responsible for managing the `_Can-*` permissions (i.e., access controls) on tables and fields including the database meta-schema. For more information, see *OpenEdge Getting Started: Core Business Services - Security and Auditing* and *OpenEdge Getting Started: Identity Management*.

Tables used from SQL only

An SQL database administrator (DBA) is a person assigned a `sysdbauth` record in the database. SQL DBAs have access to all meta data and data in the database. To support internal schema caching, every OpenEdge database begins with a DBA defined as "sysprogress." However, OpenEdge restricts the use of "sysprogress."

When you create an OpenEdge database using the PROCOPY or PRODB commands, and the database does not have any `_User` records defined (from the source database, for example), then a DBA is automatically designated with the login ID of the person who creates the database. This person can log into the database and use the `GRANT` statement to designate additional SQL DBAs, and use the `CREATE USER` and `DROP USER` statements to add and delete user IDs. When creating users, this DBA can also specify users as SQL-only users, who can only access the database through SQL.

Tables used from both ABL and SQL

When you create an OpenEdge database using the PROCOPY or the PRODB command, for ABL you have unrestricted access to the database. All users are security administrators until you designate users (at least two are recommended) as security administrators.

For SQL, your OS login ID is used to automatically designate you as a SQL DBA, but only if there are no predefined `_User` table records or DBAs (other than "sysprogress") in the source database. Once you create SQL users in the `_User` table, you must use the GRANT statement to give at least one of these users the DBA privileges in order to access SQL as a DBA.

Then:

- For ABL, you should use the database administration tools to designate an (preferably, more than one) ABL Security Administrator user ID and use it to grant access privileges.
- For SQL, you must **as your first task** create an SQL DBA account as the first user account. Use the `GRANT` statement to designate additional SQL DBAs, and use the `CREATE USER` and `DROP USER` statements to add and delete user IDs.

Note: The database tables are fully usable and manageable from ABL and SQL. ABL provides full interoperability for the tables it can access.

Connection security

Connection security ensures that only authorized users connect to a database. To establish connection security, you must establish OpenEdge-supported user accounts, configure appropriate OpenEdge security domains for these users, and assign one or more users to the security administrator role. Once connection security is established, users who are not security administrators have limited access to the **Security** menu and some other database options; they can only modify their own passwords and run a quick user report.

Note: OpenEdge supports data privacy for client-server communications with Secure Sockets Layer (SSL) connections. For more information, see Starting database brokers for SSL connections on page 79.

Designating valid users in the _User table accounts

When the database is first created, any ABL user can define users in the `_User` table accounts; access is not restricted. However, once you add security administrators, only those security administrators can add users to the `_User` table accounts. For information on defining security administrators, see Designating a security administrator on page 175.

To add a user to the _User table accounts:

1. Access the graphical Data Administration tool or the character Data Dictionary and choose **Admin** > **Security** > **Edit User List**. The **Edit User List** dialog box appears.
2. Click **Add**. The **Add User** dialog box appears.

Note:

Remember that user IDs and user names are not case sensitive, but passwords are case sensitive. Anytime you want to cancel your entries, click **Cancel** or press **END-ERROR**.

---

3. Enter the user ID (the effective user name) in the **User ID** field.
4. Click **Select Domain** to select the domain name for the specified user ID. The **Select Domain** dialog appears.
5. Select a domain from the list of domains, and click **OK**. Note that the selected domain must be configured for authenticating users in the `_User` table accounts.
6. Enter a personal name (typically the first and last given name) in the **User Name** field. This allows you to keep track of the person assigned to that user ID. Since the value in this field is not used for security purposes, you can enter any text in this field.
7. Enter the password in the **Password** field.
8. Enter `no` or `yes` in the **SQL Only** field. A `yes` value for the **SQL only** field makes the user visible to the OpenEdge SQL, and transparent to OpenEdge ABL (meaning that the user is unable to log in from ABL using the user ID).
9. Click **OK**. You are prompted to verify the password.
10. Enter the password again. If you successfully enter the same password, you will see the user record added. If you enter a different password, no user record is created.
11. To add another user record, click **Add** again. A new set of fields appears on the screen.

Note: You cannot change a **User ID** field or **Password** field from the **Edit User List** dialog box once you have added the user. However, you can change a user ID or password by deleting and then re-creating the user record. **Modify** only allows you to make changes to the **User Name** field.

After a user record is added to the `_User` table accounts, the user can connect to that database using the password you assigned. Users can then change the assigned passwords to their own private passwords. See Changing a password for database users on page 176 for instructions.

If users forget their passwords, you can delete their records from the `_User` table accounts and re-create new user records for them.

Designating a security administrator

Once you have completed configuring the database's user authentication, you can designate at least one user, and preferably more than one user, as security administrator. The security administrator is responsible for maintaining the security information in the database. Typically, the database administrator also serves as a security administrator. An ABL security administrator can manage only ABL client security (not SQL client security). Although it is not a requirement that you establish a security administrator, it is a best practice.

Security administrators can also be users defined outside of the `_User` table accounts. In fact, if you never define a user in the `_User` table accounts, all your security administrators can all be users defined in user accounts outside the database.

Once you designate security administrators, only they can access the following options on the **Security** menu:

- **Edit User List**
- **Change/Display Data Security**
- **Security Administrators**
- **Disallow Blank Userid Access**

In addition, only security administrators can access the **Database Options** function on the **Admin** menu.

All users can access the other two options in the **Security** menu: **Change Your Password** and **Quick User Reports**. Designating security administrators does not limit other users' ability to create new tables or fields in the database. In addition, a security administrator cannot change a user's password; only the user of the account can change the password.

You designate security administrators with the **Security Administrators** dialog box. Access the dialog box by choosing **Admin** > **Security** > **Security Administrators**, or by choosing **CallAdmin** from the **Edit User List** or **Edit Data Security** dialog boxes in the character Data Dictionary.

Entering user IDs for security administrators

Use the following procedure to enter user IDs for security administrators:

1. Choose **Admin** > **Security** > **Security Administrators**. The **Security Administrators** dialog box appears.
2. Enter valid user IDs.

   You can enter many user IDs here, but you must include your own user ID. Otherwise, the following error message appears:

   ```
   You cannot change a security field to exclude yourself.
   ```

   Use commas, not spaces, to separate user IDs. If you use spaces in the string, they will be accepted as part of the User ID.

3. When you are done entering user IDs, choose `OK`. You are prompted to verify the security administrator entries.
4. Click **Yes** to save the entries or **No** to return the **Security Administrators** dialog box. If you click **Yes**, you are returned to the main menu, and the specified user IDs are stored as security administrators for the database.

Deleting a user from the _User table accounts

Only security administrators can delete users from the `_User` table accounts.

To delete a user from the _User table accounts:

1. Choose **Admin** > **Security** > **Edit User List** from the graphical Data Administration tool or the character Data Dictionary. The **Edit User List** dialog box appears.
2. Select the user you want to delete, then click **Delete**. You are prompted to verify that you want to remove that user record. You cannot delete your own record until all other existing user records are deleted.
3. Verify that you want to delete the user record.

Note:

If you delete all user records from the `_User` table accounts, you are prompted to confirm that you want to remove all security restrictions for the database. If you verify the deletions, all users have security administrator privileges when they connect to the database. If you choose **No**, you must add one or more users to the `_User` table accounts before you can exit to the main window.

Changing a password for database users

Users can change their own passwords; they do not need security administrator privileges.

To change your password:

1. Log in as the account owner.
2. Choose **Admin** > **Security** > **Change Your Password** from the graphical Data Administration tool or the character Data Dictionary. You are prompted to enter your new password.
3. Enter your new password. Remember that passwords are case sensitive. You are prompted to verify the new password.

As security administrator, you can change a user's user ID or password, but only by deleting and then re-creating the user record. This way, users cannot be locked out of a database if they forget their user IDs or passwords.

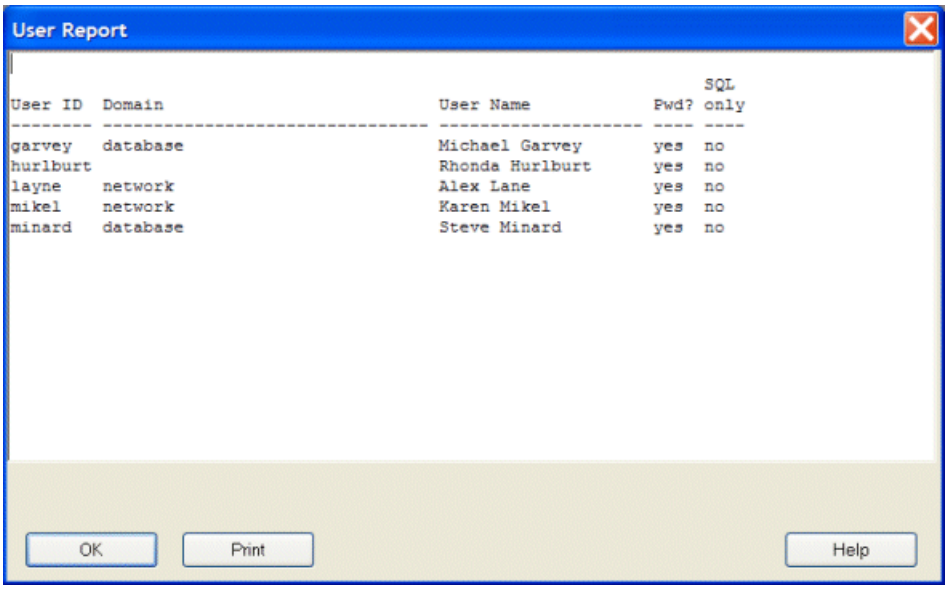
Caution: Do not try to bypass the Data Dictionary, Data Administration tool, or the Database Administration Console of OpenEdge Management or OpenEdge Explorer to modify passwords. You might lock yourself out of the database.

Running a user report

All users can display and print a user report from the **Quick User Report** option on the **Security** menu. In addition, security administrators can run a user report by choosing **Report** from the character **Edit User List** and **Edit Data Security** dialog boxes. User reports include a list of user IDs and user names and indicate which users IDs are protected by passwords.

To generate a user report:

1. Choose **Admin** > **Security** > **User Report** from the graphical Data Administration tool or the character Data Dictionary. A report similar to the following is generated:



2. Click **Print** to output the report to a printer or file. The **Print Options** dialog box appears.
3. Specify the report destination. You can print the report, or save it to a file. If you want to save the report to a file, specify the filename and whether to append the report to an existing file.
4. Specify the page length. then click **OK**. The report is directed to the specified output destination.

Schema security

Schema security ensures that only authorized users can modify table, field, and index definitions. To establish schema security, use the Data Dictionary Freeze/Unfreeze utility to lock or freeze table, field, and index definitions in the database so that unauthorized users cannot make any modifications to them. Schema modifications change file time stamps, or cyclic redundancy check (CRC) values, making all procedures that reference the database invalid. After you freeze a database, no user can make any changes to it.

Freezing and unfreezing schema

You can unfreeze frozen tables to make changes, such as adding or deleting fields or indexes or modifying field or index definitions.

To freeze or unfreeze a table:

1. Choose **Utilities** > **Freeze/Unfreeze** from the graphical Data Administration tool or the character Data Dictionary. The tables defined for the working database are listed alphabetically.
2. Choose the table you want to freeze or unfreeze. The **Freeze/Unfreeze Table** dialog box appears.
3. Specify the table status, then choose **OK**.

As security administrator, you can control which users have authority to freeze or unfreeze database tables. To freeze or unfreeze tables, the user must have Can-write access to the `_File` table and Can-write access to the `_File_Frozen` field.

Operating systems and database security

Each operating system provides security measures that you can use to protect your database.

You might want to allow access to the database from within an ABL session, but not from the operating system command level. On UNIX, you can use the operating system permissions to define security for the application database file.

Protecting the application database file using operating system permissions prevents users from accessing the database outside of ABL, or with `CONNECT` statements from an ABL session, but it does not prevent anyone from starting single-user or multi-user ABL sessions. It also does not prevent anyone from accessing the database through a server.

To protect the database file with operating system permissions, create the database under a particular user ID. Change the permission of the database table by entering the following UNIX command:

```
chmod 600 db-name
```

This permission ensures that only the owner of the database file can access the file directly.

9

Auditing

As part of the OpenEdge core business services strategy, auditing provides an efficient and scalable mechanism to produce an audit trail of access to an application's operations and data.

For details, see the following topics:

- Auditable events
- Auditing states
- Enabling and disabling auditing
- Auditing tables
- Archiving audit data
- Reconfiguring audit areas
- Auditing impact on database resources
- Auditing impact on database utilities
- Utility modifications

Auditable events

Auditing in OpenEdge is highly customizable. You can coarsely track events in your application at a very high level, and you can finely track changes to an individual field at a low level. A defined set of tracked events is called a *policy*. Policies are created using the Audit Policy Maintenance. For details on creating and maintaining policies, see the Audit Policy Maintenance online Help.

There are many types of activity that can be audited. The following sections describe the categories of auditable events:

- Audit events on page 180
- Security events on page 180
- Schema events on page 181
- Data events on page 181
- Administration events on page 181
- Utility events on page 181
- User events on page 182
- Application events on page 182

Note: To ensure you have all the current audit events in your database meta-schema, run PROUTIL UPDATESCHEMA. See PROUTIL UPDATESCHEMA qualifier on page 777 for details.

Audit events

Audit events record changes to the auditing process and policy definitions, and the manipulation of the audit trail data. Specific events include:

- Enabling and disabling a database for auditing
- Creating, updating, and deleting of audit policy records
- Runtime updating of a policy
- Dumping and loading of audit data
- Dumping and loading of audit policies
- Archiving audit data
- Removing audit administrator permissions from the last authorized audit administrator

Security events

Security events record changes to users, roles, authentication systems, and domains. Security events also record changes to SQL privileges. Specific events include creating, updating and deleting any of the following:

- A user account
- A SQL DBA
- An authentication system
- An authentication domain
- A role definition
- A role assignment
- A SQL table privilege

- A SQL column privilege
- A SQL sequence privilege

Schema events

Schema events record changes to your database schema. Specific events include creating, updating and deleting any of the following:

- A table
- A table trigger
- A field in a table
- A field trigger
- An index on a table
- A field in an index
- A sequence
- A database property (creating and updating only)

Data events

Data events record changes to the data in your database. The three specific auditable events record the create, update, and delete of a record. You can track database events by table or by field. When auditing events at the field level, table level events must also be defined. Field events will take precedence over table events.

Administration events

Administration events record the start and stop of your database.

Utility events

Utility events record when database utilities of interest are run against your database. Specific events include:

- Binary dump and load
- Table move
- Index move
- Index check
- Index rebuild
- Truncate area
- Index fix
- SQL dump and load
- Text-based dump and load

User events

User events record user identities, login activity, and database connections. Specific events include:

- Database user identity set
- Database user identity failed
- ABL user login successful
- ABL user login failed
- ABL user logout
- ABL user database connect
- ABL user database disconnect
- SQL user login successful
- SQL user login failed
- SQL user logout
- SQL user database connect
- SQL user database disconnect

Application events

Pre-defined application events track application context and audit event group. Most application events are user defined and added to your application programmatically at critical locations in the flow of execution where database events are not occurring. Event groups are used to programmatically join a series of events into a logical unit. For more information on application events and event groups, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

Auditing states

There are three possible auditing states for your database:

- **Enabled** — In an enabled state, access to the database is restricted and tracked by auditing:
  - Clients and servers from Release 10.0B and prior **cannot** start or connect to the database.
  - Audit privileges of clients from Releases 10.1A and forward are validated before granting access to the audit data, protecting your data from unauthorized access.
  - Database and application activity is audited according to active auditing policies.
- **Disabled** — In a disabled state, access to the database is **not** restricted by auditing:
  - Clients and servers from Release 10.0B and prior **can** start and connect to the database.
  - Access to the audit data by clients from Release 10.1A and forward is **disallowed**.
  - Audit privileges of clients from Release 10.1A and forward are not validated.
  - Database activity is **not** audited.

- **Deactivated** — In a deactivated state, access to the database is restricted by auditing:
  - Clients and servers from Release 10.0B and prior **cannot** start or connect to the database.
  - Access to the audit data by clients from Release 10.1A and forward is **disallowed**.
  - Audit privileges of clients from Release 10.1A and forward are not validated.
  - Database activity is **not** audited.

Enabling and disabling auditing

Auditing is enabled in an OpenEdge database with the PROUTIL utility. The first time that auditing is enabled for a new database, the auditing meta-schema tables for audit policies, events, and data are created for the database. See Auditing tables on page 185 for descriptions of the tables and indexes.

When enabling a database for auditing, you must specify an area for the audit tables. Do not place the auditing tables in the schema area. It is best to create a new area exclusively for auditing. You can also specify an area for the audit table indexes. If you anticipate generating large volumes of audit data, you may achieve better performance by separating the data and indexes. You can also choose to deactivate the non-unique auditing indexes. This could provide you with a performance boost. Deactivated indexes can be activated at a later time using PROUTIL IDXBUILD.

Enabling auditing

The following steps detail the process for enabling a database for auditing. Ensuring the security of your audit data requires that the audit data tables be **empty** when you enable auditing. When you enable your database for auditing, your database can be in one of two possible states: auditing is disabled (or has never been previously enabled) or auditing is deactivated.

To enable a database for auditing that was not previously enabled:

1. Create a structure file (`.st`) defining an area for your audit tables, and optionally an area for your audit indexes.
2. Add the areas to your database using PROSTRCT ADD. For more information on creating a structure file, and using PROSTRCT, see Maintaining Database Structure on page 325.
3. Enable your database with PROUTIL ENABLEAUDITING. The following command enables auditing for a database named `sample-db`, and places the auditing tables in an area named "Audit Area", the auditing indexes in an area named "Audit Indexes", and deactivates the non-primary indexes:

```
proutil sample-db -C enableauditing area "Audit Area"
           indexarea "Audit Indexes" deactivateidx
```

   Successfully enabling the database results in the following message:

```
Auditing has been enabled for database sample-db. (12479)
```

Once you have enabled a database for auditing, you must define and activate an auditing policy to begin generating an audit trail. For information on auditing policies, see the Audit Policy Maintenance online Help.

Re-enabling auditing from a deactivated state

To re-enable a database for auditing from a deactivated state:

1. Ensure that your audit data tables are empty.
2. Enable your database with PROUTIL ENABLEAUDITING. The following command enables auditing for a database named `sample-db`, and places the auditing tables in an area named "Audit Area", the auditing indexes in an area named "Audit Indexes", and deactivates the non-primary indexes:

```
proutil sample-db -C enableauditing
```

   Successfully enabling the database results in the following message:

```
Auditing has been activated
```

For a complete discussion of the PROUTIL ENABLEAUDITING syntax, see PROUTIL ENABLEAUDITING qualifier on page 695.

Disabling auditing

Use the command PROUTIL DISABLEAUDITING to disable auditing for a database. If a database has data in the audit data tables, `_aud-audit-data` and `_aud-audit-data-value`, when the command is issued, the database is **deactivated** rather than disabled. The syntax of the DISABLEAUDITING command is:

```
proutil db-name -C disableauditing
```

The following message is displayed when auditing is deactivated:

```
Auditing was not fully disabled because auditing data tables are not empty.
(13647)
Auditing has been deactivated, no additional auditing records will be recorded.
(13649)
```

To fully disable auditing, the `_aud-audit-data` and `_aud-audit-data-value` tables must be empty. The following procedure provides the steps to disable auditing on a database that currently has auditing deactivated.

To disable auditing on a deactivated database:

1. Activate auditing with PROUTIL ENABLEAUDITING.

   Because the database is **deactivated**, issuing PROUTIL ENABLEAUDITING has the effect of **activating** auditing.

2. Empty the audit data tables.

   Use your audit archiving tools or PROUTIL AUDITARCHIVE to remove the data from the `_aud-audit-data` and `_aud-audit-data-value` tables.

Note: Not all audit records are deleted by PROUTIL AUDITARCHIVE. For example, if your audit policies audit the execution of AUDITARCHIVE, you will need to use another method to remove all the data from the audit data tables.

3. Disable auditing with PROUTIL DISABLEAUDITNG. Because the audit data tables are empty, auditing is completely disabled.

   When auditing is successfully **disabled**, the following message is displayed:

```
Auditing has been disabled for database db-name. (12490)
```

Auditing tables

Enabling your database for auditing creates seven new tables in your database meta-schema. The table below briefly describes each of the tables. The details on the fields and indexes of each of these tables are discussed in *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

Table 32: Auditing meta-schema tables

Table name	Description	Archived
_aud-audit-data	This table contains the audit data records. All events are stored in this table.	Yes
_aud-audit-data-value	This table is a child table of the _aud-audit-data table, and contains records for each field change data event.	Yes
_aud-audit-policy	This table contains named audit policies. If multiple policies are active, the aggregation of the policies is applied, and the highest level of auditing will be applied if a conflict exists between policies.	No
_aud-event	This table contains the definitions for all supported OpenEdge and user-defined audit events and their event ids. All event ids up to 32,000 are reserved. You can create custom application events with ids greater than 32,000.	Yes
_aud-event-policy	This table contains policy settings for events associated with policies.	No
_aud-field-policy	This table contains field level auditing settings associated with a named policy.	No
_aud-file-policy	This table contains table level auditing settings associated with a named policy.	No

Indexes on auditing tables

Most of the tables that support auditing have multiple indexes. When enabling auditing, you have the option of activating all the indexes, or only the primary indexes. Generally, only the primary indexes should be activated in a production database. The non-primary indexes are used for reporting and should be enabled in your archive database. The table below lists the indexes for each of the auditing-related schema tables. See *OpenEdge Getting Started: Core Business Services - Security and Auditing* for details on the components of each index.

Table 33: Indexes for auditing schema tables

Table	Index name	Index type
_aud-audit-data	_AppContext-Id	–
	_Audit-time	–
	_Connection-id	–
	_Data-guid	Primary, Unique
	_Event-context	–
	_Event-group	–
	_EventId	–
	_Userid	–
_aud-audit-data-value	_Field-name	–
	_Continuation-seq	–
_aud-audit-policy	_Policy-active	–
	_Policy-desc	–
	_Policy-guid	Primary, Unique
	_Policy-name	Unique
_aud-event	_Event-desc	–
	_Event-id	Primary, Unique
	_Event-name	–
_aud-event-policy	_Event-guid	Primary, Unique
	_Event-id	–
_aud-field-policy	_Field-name	Primary, Unique
	_File-field-owner	–

Table	Index name	Index type
_aud-file-policy	_Create-event-id	–
	_Delete-event-id	–
	_File-owner	–
	_Guid-file-owner	Primary, Unique
	_Read-event-id	–
	_Update-event-id	–
_Client-session	_Auth-time	–
	_Db-guid	–
	_Session-uuid	Primary, Unique
	_Userid	–
_Db-detail	_Db-guid	Primary, Unique

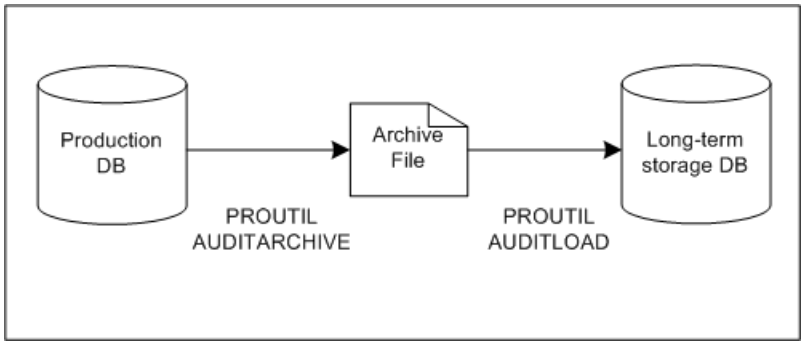
Archiving audit data

Once you have enabled auditing for your database, you are likely to generate a large quantity of audit data. To keep your production database from growing too large too quickly, and for greater reporting capabilities (remember, Progress recommends only minimal indexing of the audit tables in your production database), you are going to want to archive your audit data to a long-term storage database. You can store archived audit data from multiple databases in your long-term storage, and with all indexes activated, you can more quickly access the records you need.

Audit archive process

Archiving your audit data is a multi-step process that varies depending on your environment. You archive data to an archive file with PROUTIL AUDITARCHIVE, and you load data from the archive file with PROUTIL AUDITLOAD, as shown in the following figure.

Figure 17: Audit archive process



Note: Audit archive files created with OpenEdge Version 12.0 can only be loaded by the OpenEdge Version 12.0 audit load utility. They cannot by loaded with prior versions of the audit load utility. Audit archive files created by audit archive utility versions prior to OpenEdge Version 12.0 can be loaded by the OpenEdge Version 12.0 audit load utility.

The user performing the audit archive must have the Audit Archive privilege for the source database; the user performing the audit load must have the Audit Data Archiver privilege for the target database. For complete utility details, see PROUTIL AUDITARCHIVE qualifier on page 640 and PROUTIL AUDITLOAD qualifier on page 643.

Basic archive process

The most basic archive process is to archive and delete all the audit data from your production database and load the data into your archive database. If your production database is called `prod-db`, and your archive is `arch-db`, these are the basic steps:

1. Archive the data:

```
proutil prod-db -C auditarchive
```

   This command archives all the audit data currently in `prod-db` to your current working directory, and deletes all the audit data from `prod-db`. The audit archive file is named `prod-db.abd`.

2. Load the data:

```
proutil arch-db -C auditload /usr1/prod-db-dir/prod-db.abd
```

   This command loads all the audit data in the archive file `/usr1/prod-db-dir/prod-db.abd` into the `arch-db` database.

Archive process with seal verification

A more cautious approach to archiving your audit data is to verify the data seals, and to confirm that the archive and load are successful prior to deleting the data from your production database. If your production database is online during this archive procedure, you need to specify a date or date range for the audit archive. Specifying a date or date range guarantees that you will archive the same records in Step 1 and Step 3 below. If you specify one date, all audit data records up to the specified date are archived; if you specify two dates, all records between the two dates (inclusive) are archived. If your production database is called `prod-db`, and your archive is `arch-db`, and you want to archive all data prior to midnight on August 31, 2005 EDT, these are the basic steps:

1. Archive the data, checking the data seal as you archive:

```
proutil prod-db -C auditarchive "08-31-2005 23:59:59.999-04:00"
 -checkseal -nodelete -directory /usr1/audit_archives
```

This command archives all the audit data currently in `prod-db` to the file `/usr1/audit_archives/prod-db.abd`, but does not delete any of the audit data from `prod-db` because of the `-nodelete` qualifier. The `-checkseal` qualifier, directs the archive process to verify the data seal of each audit data record prior to archiving it.

2. Load the data:

```
proutil arch-db -C auditload /usr1/audit_archives/prod-db.abd -checkseal
```

This command loads all the audit data currently stored in the `/usr1/audit_archives/_aud-audit-data.abd`. file into the `arch-db` database, checking the seal of each audit data record prior to loading it into the archive database.

3. Delete the data:

```
proutil prod-db -C auditarchive "08-31-2005 23:59:59.999-04:00"
             -directory /dev/null
```

This command deletes the audit data archived from the database in the first step of this general process, and does not produce a new archive file.

Reconfiguring audit areas

Over time, your auditing requirements may change, and as a result, you may find that the original layout of your audit data and index areas, specified when you enabled auditing with PROUTIL ENABLEAUDITING, and their subsequent growth is unsatisfactory.

PROUTIL AUDITRECONFIG provides Audit Data Archivers the ability to reconfigure audit data and index areas while auditing is enabled. You can move the audit data tables and indexes to new areas, or you can purge and reuse your current areas.

When reconfiguring your audit data and index areas, you can:

- Split or combine index and data areas
- Set new records per block and blocks per cluster values in new areas
- Purge all data from existing audit areas by specifying the existing areas in the PROUTIL AUDITRECONFIG command

---

Note: You can reclaim disk space by deleting your old audit data or index areas if they are obsolete after configuring new areas.

AUDITRECONFIG destroys all data in your current audit data areas; however only reconfiguring your index area does not destroy data. Preserve your auditing data by archiving with PROUTIL AUDITARCHIVE prior to executing PROUTIL AUDITRECONFIG. See Archiving audit data on page 187 for information on archiving your audit data.

The following sections provide general steps for PROUTIL AUDITRECONFIG uses:

- Split one audit area into separate data and index areas on page 190
- Combine audit data and index areas into one area on page 190

- Move existing audit data and indexes to new areas on page 191
- Purge data in existing audit data areas on page 192

Split one audit area into separate data and index areas

Suppose that when you first enabled your database for auditing, you created one area for both the data and indexes. Now you want to move the audit indexes into a separate area. Use the following steps as a guide:

To split one audit area into separate data and index areas:

1. Create a structure file to define a new area to hold the audit indexes. See Maintaining Database Structure on page 325 for information on evaluating database structure and using PROSTRCT to modify the physical structure of your database.
2. Add the new area to your database with PROSTRCT ADD.
3. Reconfigure your index areas.

   Use the following syntax:

```
proutil db-name -C auditreconfig indexarea new-index-area-name
```

   Successful completion of the reconfiguration produces the following messages:

```
Auditreconfig started. (16973)
Reconfiguring indexes ... (16978)
Auditreconfig completed. (16975)
```

See PROUTIL AUDITRECONFIG qualifier on page 644 for the complete syntax.

Combine audit data and index areas into one area

Suppose that when you first enabled your database for auditing, you created separate areas for the data and indexes. Now you want to move the audit indexes into the same area as the data. Use the following steps as a guide:

To combine audit areas:

1. Suppose your existing audit data area is called "`Audit Data`" and your index area is called "`Audit Indexes`". Reconfigure your index area into the existing data areas.

   Use the following syntax:

```
proutil db-name -C auditreconfig indexarea "Audit Data"
```

Note: See PROUTIL AUDITRECONFIG qualifier on page 644 for the complete syntax.

Successful completion of the reconfiguration produces the following messages:

```
Auditreconfig started. (16973)
Reconfiguring indexes ... (16978)
Auditreconfig completed. (16975)
```

2. Remove the old audit index area with PROSTRCT REMOVE.

Move existing audit data and indexes to new areas

Suppose that when you first enabled your database for auditing, you created two area for the data and indexes. Now you find they are not properly sized and you want to move the audit data and indexes into areas with different attributes. Use the following steps as a guide:

To split one audit area:

1. Create a structure file to define two new areas to hold the audit data and indexes. Change the records per block and blocks per cluster values as needed. See Maintaining Database Structure on page 325 for information on evaluating database structure and using PROSTRCT to modify the physical structure of your database.
2. Add the new areas to your database with PROSTRCT ADD.
3. Archive any existing audit data with PROUTIL AUDITARCHIVE.

   The syntax is:

   ```
   proutil db-name -C auditarchive
   ```

   If you archive the data with no parameters, the data in your audit tables is deleted after it is archived. See PROUTIL AUDITARCHIVE qualifier on page 640 for the complete syntax.

4. Reconfigure your auditing areas with PROUTIL AUDITRECONFIG.

   The syntax is:

   ```
   proutil db-name -C auditreconfig tablearea new-table-area-name
               indexarea new-index-area-name
   ```

   See PROUTIL AUDITRECONFIG qualifier on page 644 for the complete syntax.

5. PROUTIL AUDITRECONFIG prompts you to confirm your data is archived.

   The prompt appears as shown:

   ```
   The operation will delete all auditing data records.
   You must archive auditing data first.
   Have you archived the data already and do you want to continue? (Y/N) (16987)
   ```

6. Enter **Y** to proceed with the reconfiguration.

   Successful completion of the reconfiguration produces the following messages:

```
Auditreconfig started. (16973)
Reconfiguring tables and indexes ... (16944)
Auditreconfig completed. (16975)
```

7. Remove the old audit data and index areas with PROSTRCT REMOVE.

Purge data in existing audit data areas

Suppose that you want to purge your database of all auditing data and restart recording audited events in the area you are currently using. Use the following steps as a guide:

To reconfigure your audit data area:

1. Archive any existing audit data with PROUTIL AUDITARCHIVE.

   The syntax is:

```
proutil db-name -C auditarchive
```

   If you archive the data with no parameters, the data in your audit tables is deleted after it is archived. See PROUTIL AUDITARCHIVE qualifier on page 640 for the complete syntax.

2. Reconfigure your existing audit data area.

   Use the following syntax (suppose the existing audit data area is named "Audit Data"):

```
proutil db-name -C auditreconfig tablearea "Audit Data"
```

3. PROUTIL AUDITRECONFIG prompts you to confirm your data is archived.

   The prompt appears as follows:

```
The operation will delete all auditing data records.
You must archive auditing data first.
Have you archived the data already and do you want to continue? (Y/N)
(16987)
```

4. Enter **Y** to proceed with the reconfiguration.

   Successful completion of the reconfiguration produces the following messages:

```
Auditreconfig started. (16973)
Reconfiguring tables ... (16979)
Auditreconfig completed. (16975)
```

See PROUTIL AUDITRECONFIG qualifier on page 644 for the complete syntax.

Auditing impact on database resources

Auditing has the potential to be extremely resource-intensive. A new record is written to the `_aud-data table` every time an event occurs. An additional record or records is written to the `_aud-data-value` table every time an audited field data change event occurs. Depending on the level of auditing specified, you could be storing both the original and the changed values of a field every time it is updated.

Auditing impact on database utilities

The need to guarantee the integrity of the audit trail requires that only authenticated users have access to the auditing data and policy tables. To ensure the guarantee, the use of certain database utilities is restricted to authenticated users.

Auditing incorporates user security into the database utilities. This addition implements the three primary aspects of security: authentication, authorization, and auditing. It is necessary to determine who is using the utility, if that user is allowed to perform the indicated action on the specified resource, and record the details of the action.

Not all database utilities require user security. The database utilities that do require this additional security meet one of the following criteria:

- The utility alters the status of database auditing
- The utility can alter the audit data or policies
- The utility can alter the database areas defined for audit data and indexes

Access to the utilities is acquired when the user is granted a privileged auditing role. The user is identified by the database `_User` table, or the operating system login (effective process user-id.) Depending on the utility, the user must be granted the Audit Administrator role or the Audit Data Archiver role. For information on the definitions of these roles and how to grant auditing roles, see *OpenEdge Getting Started: Core Business Services - Security and Auditing* or the Data Administration online Help.

Identifying the privileged user

Two options exist for user identification, the database `_User` table or the operating system login (effective process user-id). Based on the choice of user identification, passing user information to the utilities differs as described.

_User table

If the `_User` table is established for the database, `-userid` *`username`* must be included to specify a privileged user to authenticate. The specified user, *`username`*, must be defined in the `_User` table and have the required role. The specified password, *`passwd`*, must be the password defined in `_User` table to match *`username`*. The password can be specified with the `-password` *`passwd`*qualifier or if omitted, the utility will prompt for the value. The value of *`passwd`* can be in clear text format or encrypted. For details on specifying an encrypted password, see Specifying encrypted passwords on page 195.

If you have defined the user "Mike" in the `_User` table with a password of "guru" for the database `auditexampledb`, and assigned "Mike" the Audit Data Archiver role, then executing the protected PROUTIL AUDITARCHIVE utility for the database would use one of the following formats:

- Clear text password:

```
$ proutil auditexampledb -C auditarchive -userid Mike -password guru
```

- Prompt for password:

```
$ proutil auditexampledb -C auditarchive -userid Mike
OpenEdge Release 10.1A as of Sat Dec 17 09:56:25 EST 2005
password: ****
```

At the password prompt, the password "guru" must be typed before the AUDITARCHIVE runs.

- Encrypted password:

```
$ genpassword -password guru
37273d32
.
.
.
$ proutil auditexampledb -C auditarchive -userid Mike
            -password oech1::37273d32
```

First, you must encrypt the password using `genpassword`. Then, when you run the AUDITARCHIVE utility (presumably at a later time), specify the encrypted password in the command.

Local operating system login

If the `_User` table is not used in the database, the local operating system login (effective process user-id) identifies the privileged user. This user must be granted at least the Audit Administrator role. Once the appropriate roles are granted to the user, no further action is required. The utilities are to trust the operating system user verification, and the user can run the utilities without specifying any additional command-line parameters.

Optionally, a local operating system user-id can be specified on the utility command line by adding the `-userid` *`username`* qualifier. The consequence of adding `-userid` is that it requires a password. The password can be specified with the `-password` qualifier. If the `-password` qualifier is not specified, the utility will prompt for the password to be entered. For the local operating system user, the password for the enhanced utilities is not the operating system login password. The utilities require the encrypted database MAC key (DB Pass key) for the password. The database MAC key is stored in the `_db-detail` table of the database in the `_db-mac-key` field, and is set through the Data Administration tool. For details on setting the DB Pass Key, see *OpenEdge Getting Started: Core Business Services - Security and Auditing* or the Data Administration online Help. For details on specifying encrypted passwords, see Specifying encrypted passwords on page 195.

If your operating system login is "sysdba", and you have not established the `_User` table, and you have assigned "sysdba" the Audit Data Archiver role for the database `auditexampledb`, then executing the protected PROUTIL AUDITARCHIVE utility for the database would use one of the following formats:

- Trust the operating system authentication:

```
$ proutil auditexampledb -C auditarchive
```

- Require DB Pass Key on command line:

```
$ genpassword -password ultra_secret_password
253e3b35331a203633202a330d3532202325203536
.
.
.
proutil auditexampledb -C auditarchive -userid sysdba
      -password oech1::253e3b35331a203633202a330d3532202325203536
```

For this example, assume that the DB Pass Key is "utlra_secret_password". First, you must encrypt the DB Pass Key using `genpassword`. Then, when you run the AUDITARCHIVE utility (presumably at a later time), specify the encrypted DB Pass Key in the command.

- Prompt for DB Pass Key:

```
$ proutil auditexampledb -C auditarchive -userid sysdba
OpenEdge Release 10.1A as of Sat Dec 17 09:56:25 EST 2005
password: **********************
```

At the password prompt, the DB Pass Key must be typed before the AUDITARCHIVE runs. The password value is obfuscated as it is typed, and can be either the clear text value, or the encrypted value, provided it has the proper encryption prefix.

Specifying encrypted passwords

Encrypted passwords enhance security, particularly when generating maintenance scripts that contain calls to the utilities with passwords.

The format of an encrypted password for the database utilities is `oech1::`*`password`*. It consists of a five letter prefix, followed by a separator, followed by the encrypted password. The table below details the components of an encrypted password.

Table 34: Encrypted password components

Component	Description
`oec`	Abbreviation identifying the encryption algorithm, OpenEdge crypto
`h1`	A hex-binary encoded value identifying the encryption algorithm key-size as 1
`::`	Separator
`password`	Encrypted password; the encryption is performed with the algorithm specified in the prefix

Encrypt clear text passwords with the utility `genpassword`. See *OpenEdge Getting Started: Installation and Configuration* for a detailed description of `genpassword`.

Utility modifications

The utilities enhanced with user security either access protected auditing data or alter the state of auditing. This section defines the protected data and enumerates the impacted utilities in the following sections:

- Protected tables on page 196
- Protected areas on page 196
- Enhanced utilities on page 196
- Expanded PROUTIL syntax on page 197

Protected tables

Protected tables are those database tables used in the auditing process and whose manipulation is restricted to privileged users. There are two types of protected tables: audit data tables and audit policy tables. The table below lists the protected tables.

Table 35: Protected tables

Table name	Table type
_aud-audit-data	Data
_aud-audit-data-value	Data
_client-session	Data
_aud-event	Policy
_aud-event-policy	Policy
_aud-audit-policy	Policy
_aud-file-policy	Policy
_aud-field-policy	Policy

Protected areas

Protected areas contain audit data and indexes. These areas are specified when auditing is enabled.

Enhanced utilities

The enhanced utilities operate on the protected tables or protected areas, or alter the status of auditing. For some utilities, the operation is unconditionally denied. For others, the operation can run if the user has the required privilege. The table below lists the protected utilities. The table also details the impacted objects and required privilege if applicable, and the access restrictions.

Table 36: Protected utilities

Operation	PROUTIL qualifier	Impacted protected object	Required privilege	Restriction
Bulk load	`bulkload`	Protected tables	–	Denied to all users
Binary load	`load`	Protected tables	–	Denied to all users
Binary dump	`dump dumpspecified`	Protected tables	–	Denied to all users
Index fix (delete record)	`idxfix`	Protected tables	Audit Archiver	Denied to all except Audit Archiver
Truncate area	`truncate area`	Protected areas	—	Denied to all users
Archive audit data	`auditarchive`	Protected tables	Audit Archiver	Denied to all except Audit Archiver
Reconfigure audit area	auditreconfig	Protected tables and indexes	Audit Archiver	Denied to all except Audit Archiver
Load audit data	`auditload`	Protected tables	Audit Archiver	Denied to all except Audit Archiver
Disable auditing	`disableauditing`	–	Audit Administrator	Denied to all except Audit Administrator

Expanded PROUTIL syntax

For the enhanced utilities that allow access to properly privileged and authenticated users, the syntax of PROUTIL is expanded to include the user id and password as follows:

Syntax

```
PROUTIL dbname -C qualifier [qualifier-parameters]
        [ -userid username [-password passwd] ]
```

10

Transparent Data Encryption

Transparent Data Encryption, as part of an overall security strategy, provides for data privacy while the data is "at rest" in your OpenEdge database, regardless of the location of the database and who has a copy of it. The Transparent Data Encryption product must be installed in conjunction with an Enterprise Database license.

For details, see the following topics:

- Overview
- Enabling Encryption
- Creating encryption policies
- Encrypting your existing data
- Running with Transparent Data Encryption enabled
- Maintaining Transparent Data Encryption
- Disabling encryption
- OpenEdge SQL support for Transparent Data Encryption

Overview

Controlling access to private data while "at rest," that is, stored on disk inside your database, is the core of OpenEdge Transparent Data Encryption. OpenEdge combines various cipher algorithms, various encryption key lengths, secure storage of encryption keys, and user access controls to your encryption keys to ensure that your data's encryption cannot be reversed by anyone other than those granted access.

Each encrypted database has a single, unique Database Master Key (DMK). The DMK is created and managed by your database administrator, and stored in your database key store, which is separate from your database. Your *key store* is an independent and secure entity that provides secure storage of data encryption keys and controls access in the form of user accounts.

Encryption of your database objects is managed through encryption policies. You define which objects are encrypted and the encryption cipher for the object. Policies are stored in your database in a designated Encryption Policy Area. Object policies use virtual data encryption keys derived from your DMK and the specified cipher. The encryption key for each encrypted database object is unique.

For an in-depth discussion of the concepts surrounding Transparent Data Encryption, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

Before you start

It is easy to get started with Transparent Data Encryption, but before you start, you should understand the following:

- Know what objects in your database need to be encrypted.

  OpenEdge Transparent Data Encryption gives you the flexibility to select which objects in your database need to be encrypted. You should select the smallest set of objects that contain private data. Knowledge of your database schema is required to select the appropriate objects. You will also need to consider the indexes of the encrypted objects, based on the fields that comprise the index. If your index contains critical (private) fields of an encrypted table, you should encrypt the index.

- Decide your AI and BI encryption strategy.

  When you enable transparent data encryption, by default your BI files and AI files (if enabled) are also enabled for encryption. Progress Software Corporation strongly encourages you to encrypt your BI and AI files. Failure to encrypt your BI and AI files exposes your encrypted data in an unencrypted form in your BI and AI notes. If you decide to risk unencrypted data being exposed through your AI and BI notes, you can disable AI and BI encryption.

- Choose the cipher(s) that meet your requirements.

  OpenEdge Transparent Data Encryption supports six different ciphers. The ciphers vary in strength. You need to understand your requirements to pick the correct cipher; the stronger the cipher, the harder to break, but it also takes longer to encrypt and decrypt your data. For a general discussion of ciphers, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*. For a list of the ciphers supported for Transparent Data Encryption, see OpenEdge supported ciphers on page 203.

- Determine access to the database key store.

  To open an encryption-enabled database, you must be authenticated as able to open the database key store. The key store is created when you enable your database for encryption. For an in-depth discussion of the OpenEdge key store, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

  There are two ways to authenticate to the key store: manual start and autostart. With manual start, every time you open the database, you must supply a passphrase.

  For servers and utilities, and single-user or self-service client, you can include an additional parameter (`-Passphrase`) on the command line to indicate that you are to be prompted for a passphrase to open the key store.

  For ABL clients, the passphrase must be included in the CONNECT statement with the `-KeyStorePassPhrase` parameter. If the passphrase is authenticated, access is granted. See Running with Transparent Data Encryption enabled on page 211 for more information.

By allowing autostart, you are granting access to the key store without prompting for a passphrase. You can override the autostart authentication by including the passphrase parameter. Manual start is more secure, but impacts automated database administration (scripts); autostart does not impact scripts, but potentially gives unrestricted access to encrypted data.

What is encryptable

OpenEdge allows encryption of the following database elements:

- Type I areas
- Type II table
- Type II index
- Type II LOB

Note: You cannot encrypt the Schema area of your database. You can encrypt your audit data tables.

OpenEdge encrypts the following storage files:

- BI files (by default, but can be disabled)
- AI files (by default, but can be disabled)
- Backup files
- Binary dump (if requested)
- Audit archive (if requested)

BI and AI files are encrypted by default. You can disable their encryption, but doing so is highly discouraged because data can be exposed in an unencrypted form in your BI and AI notes. Backup files are always encrypted. Binary dump (and audit archive) files are not encrypted by default, even if the data being dumped is encrypted. You can encrypt your dump files with a password based encryption (PBE) cipher by adding an additional parameter to the command. If you intend to keep your dump or archive for an extended period of time, encrypting it may be prudent. For more information about encrypting a dump file, see PROUTIL Utility on page 629.

Data considerations

Choosing the correct data to encrypt is specific to your database and application requirements, but consider the following when selecting objects to encrypt:

- For data in a Type I area, you must encrypt the entire area, even if only one field in one table must be encrypted. Consider moving the table with the critical field, and its indexes to a Type II area, so that you can encrypt only the critical table and indexes.
- For tables in a Type II area, understand the contents of the table's indexes to determine if the index needs to be encrypted. For example, if you have an employee table with a social security number field and a separate employee number field, if the social security number field is used in an index, the index should be encrypted. However, if you have an index on the employee number field, you may determine that it is not necessary to encrypt that index.

Database key store

Your database key store is created when you enable your database for transparent data encryption using the PROUTIL ENABLEENCRYPTION command. The key store has the following main functions:

- Stores the Database Master Key (DMK) externally from the database
- Derives the database object virtual keys from the DMK
- Protects the DMK and object virtual keys from being copied
- Controls access to the key store through built-in user accounts with strong passphrase protection
- Denies access to a transparent data encryption-enabled database if the user cannot open the key store by supplying a passphrase for one of the built-in key store user accounts
- Configures opening of the key store through automated processes

A key store has two built-in user accounts: the admin account and the user account. Key store administrator privilege is required to create or change any key store value, including all aspects of encryption key generation and storage. User privilege is required to access encryption key values. You must always provide a passphrase for the key store admin account when you create the key store; the user account passphrase is optional. The passphrases for the key store user and admin accounts must be different.

Passphrases must comply with the rules described in the table below.

Table 37: Passphrase constraints

Rule	value
Minimum number of characters	8
Maximum number of characters	2048
Minimum number of numeric characters	1
Minimum number of alpha characters	2
Minimum number of punctuation characters	1
Character set	[a-zA-Z0-9]!@#$%^&*()_+-{}[]|\\,./<>?;:<space>
First character	(see Character set)
Mixed case alpha required	True
Case sensitive	True

When your key store is created, it is bound to your database, but it remains a separate entity. PROBKUP does not backup your key store. If you create a copy of your database with PROCOPY, the key store is not copied. The key store is not part of your database structure definition. If you copy an encryption-enabled database, you will not be able open the copy until you copy and rebind the key store to the copied database using the PROUTIL EPOLICY command.

The key store is separate from your database for security reasons, and you must securely back it up when you back up your database. Protecting your key store is vital to maintaining access to your encrypted database. Without your key store, you will be unable to open your database.

Compare your database and key store to your car and car keys. The key store is separate from your database because tying them together would be like leaving your car key in the door lock; your door might be locked, but the probability of someone "breaking in" is greatly increased. Also like your car, you want to have a secure back up of your car keys or key store in case the original is lost.

Configuring key store access

To open a transparent database encryption-enabled database, OpenEdge must be able to open the key store. To successfully open the key store, you must provide the correct key store passphrase. If OpenEdge cannot open the key store, then opening the database fails. You have two configuration choices for how the key store passphrase is supplied:

- **Manual mode** — Manual mode requires that you supply (type in) a key store account passphrase any time your database is opened.
- **Autostart mode** — Autostart delivers a passphrase you configure to open the key store automatically.

Manual mode is more secure, but impacts automated database administration (scripts), and requires manual intervention for every database access or invocation of an executable; Autostart mode does not impact scripts, but potentially gives unfettered access to encrypted data.

OpenEdge supported ciphers

For this release, OpenEdge supports the ciphers described in the following tables.

Table 38: Database Master key ciphers

ID	Cipher	Mode	Size	Key type
1	AES	CBC	128	binary
2	AES	CBC	192	binary
3	AES	CBC	256	binary
4	DES	CBC	56	binary
5	DES3	CBC	168	binary
6	DES	CBC	56	PBE

Table 39: Object ciphers

ID	Cipher	Mode	Size	Key type
0	NULL	NULL	–	–
1	AES	CBC	128	binary
2	AES	CBC	192	binary
3	AES	CBC	256	binary
4	DES	CBC	56	binary
5	DES3	CBC	168	binary

Enabling Encryption

When you are ready to enable your database for Transparent Data Encryption, there are two required steps, as discussed in the following sections:

- Adding an encryption policy area on page 204
- Running ENABLEENCRYPTION on page 205

Additional, optional steps, discussed in the following sections, include:

- Enabling BI file encryption after enabling encryption on page 206
- Enabling AI file encryption after enabling encryption on page 206
- Verifying enablements on page 207

Note: If you are enabling encryption on a database that is also configured for OpenEdge Replication, see 7 for additional instructions.

Adding an encryption policy area

Transparent Data Encryption requires a specific area to hold your encryption policies. To protect your policies, you cannot perform any record operations on the policy data with an ABL or SQL client. The encryption policy area has the following restrictions and characteristics:

- The area must be named "Encryption Policy Area" and the type token in your structure (`.st`) file must be an "e".
- The area number of the Encryption Policy Area must be greater than the area number of the Schema area.
- The area must be a Type II area.
- The first line defining the Encryption Policy Area in your structure (`.st`) file must contain both the area name and the area number.
- If the area definition in your structure file omits the area cluster size, the size defaults to the value of `CLUSTERSIZE_DEFAULT` (8 blocks per cluster).
- If the area number and the records per block values are omitted, the values are assigned following the rules for data areas.
- After the first definition line in the structure file, subsequent lines may omit the area name, area number, records per block, and cluster size values.

To add an Encryption Policy Area to your database:

1. Create a structure (`.st`) file describing the Encryption Policy Area.

   For example:

```
e "Encryption Policy Area":12,32;64 . f 1024
e "Encryption Policy Area":12,32;64 .
```

2. Add the Encryption Policy Area to your database with PROSTRCT ADD.

For example:

```
prostrct add mydb encrypt_policy_area.st
```

3. Create a new structure file for your database that reflects the added Encryption Policy area with PROSTRCT LIST.

   For example:

```
prostrct list mydb
```

Running ENABLEENCRYPTION

One command enables your database for transparent data encryption. The basic syntax for enabling encryption is:

```
proutil dbname -C enableencryption [-Cipher cipher-num]
           [-Autostart {user | admin}]
```

Enabling encryption performs the following tasks on your database:

- The database BI is truncated if the database is offline and the BI is not already truncated.
- The schema for encryption policy area is loaded.
- New audit events for encryption are loaded.
- The OpenEdge key store is created, and the key store creates and stores the database master key. The key store is named, *dbname*.ks, and is stored in the same directory as your *dbname*.db file.
- The master database security record is created in the encryption policies.
- A UUID for the database is set, if not already set.
- Encryption keys are generated for encrypting the database AI and BI files (unless explicitly turned off).
- Autostart is configured for the key store, if requested.
- You are prompted for passphrases:
  - The key store admin passphrase is required.
  - The key store user passphrase is optional.
  - The PBE passphrase is mandatory if you specify the PBE cipher for your key store (-Cipher 6).

By default, PROUTIL ENABLEENCRYPTION indicates that all future AI and BI notes are encrypted. If after-imaging is enabled, enabling encryption results in an extent switch. If you enable encryption while your database is online, BI notes are not encrypted; see Enabling BI file encryption after enabling encryption on page 206 for instructions on enabling your BI files for encryption. Existing AI and BI files are **not** encrypted; enabling encryption essentially sets an indicator for **future** writes. See PROUTIL ENABLEENCRYPTION qualifier on page 699 for the complete syntax.

Enabling BI file encryption after enabling encryption

If during the execution of PROUTIL ENABLEENCRYPTION, you specifically indicate that your BI files should not be encrypted, you can enable them for encryption at a later point in time in the following ways:

- You can re-issue PROUTIL ENABLEENCRYPTION and add `-biencryption enable` to the command line, as shown:

```
proutil dbname -C enableencryption -biencryption enable
```

- You can truncate your BI and add `-biencryption enable` to the command line, as shown:

```
proutil dbname -C truncate bi -biencryption enable
```

Enabling AI file encryption after enabling encryption

If you specifically indicate that your AI files should not be encrypted when you first enable encryption on your database, you can enable them for encryption at a later point in time in the following ways:

- You can re-issue PROUTIL ENABLEENCRYPTION and add `-aiencryption enable` to the command line, as shown:

```
proutil dbname -C enableencryption -aiencryption enable
```

- You can indicate that AI files are to be encrypted when you begin AI operations, as shown:

```
rfutil dbname -C aimage begin -aiencryption enable
```

  By default, if your database is enabled for encryption, enabling after-imaging attempts to enable encryption for your AI files.

- You can indicate that AI files are to be encrypted when you enable after-imaging with an online backup, as shown:

```
probkup dbnamebackupname -enableai -aiencryption enable
```

By default, if your database is enabled for encryption, enabling after-imaging enables encryption for your AI files.

Note: You can use the same commands, with `-aiencryption disable` to turn off the encryption of your AI files.

Verifying enablements

PROUTIL DESCRIBE provides information about your database and enabled features. Use PROUTIL DESCRIBE to confirm that:

- Encryption is enabled on your database
- BI encryption is enabled on your database
- AI encryption is enabled on your database

The following figure shows the output of PROUTIL DESCRIBE.

Figure 18: Output of PROUTIL DESCRIBE

```
OpenEdge Database Description

Database Name               : /usr1/dbs/sample_db
Version                     : 173.0
Block Size                  : 8192
Largest Cluster             : 64
Create Date                 : Mon Dec 10 13:04:58 2012
Last Open Date              : Mon Dec 10 13:25:24 2012
Prior Open Date             : Mon Dec 10 13:25:24 2012
Schema Change Date          : Mon Dec 10 13:05:09 2012

 Before Imaging information
  Block Size                : 8192
  Cluster Size (16K Units)  : 32
  Last Open Date            : Mon Dec 10 13:05:25 2012
  Bi Encryption             : Enabled

 After Imaging Information
  Block Size                : 8192
  Begin Date                : Mon Dec 10 13:05:25 2012
  Last AIMAGE NEW           : Mon Dec 10 13:05:26 2012
  Currently Busy Area       : 13
  Current File Number       : 2
  Ai Encryption             : Enabled

 Backup Information
  Last Full Backup Date     : Mon Dec 10 13:05:17 2012
  Last Incremental Backup   : *** Not yet performed ***

 Database Features

    ID   Feature                          Active  Details
  ----  --------------------------------  ------  -------
     1  OpenEdge Replication              Yes     Source database
     7  JTA                               Yes
     9  64 Bit DBKEYS                     Yes
    10  Large Keys                        Yes
    11  64 Bit Sequences                  Yes
    13  Encryption                        Yes
    14  Multi-tenancy                     Yes
    15  Concurrent JTA and Replication    Yes
```

For more information on PROUTIL DESCRIBE, see PROUTIL DESCRIBE qualifier on page 678.

Creating encryption policies

Encryption of your data is managed through encryption policies. When you create a policy, you specify which database object (table, index, LOB, or Type I area) to encrypt and the strength of the encryption cipher for the object. If you do not specify a cipher, the default, AES_CBC_128, is used. Creating a policy does **not** encrypt your existing data; it indicates that all future writes of the data is encrypted. See Encrypting your existing data on page 209 for instructions on encrypting your existing data.

Encryption policies are created in several ways. See one of the following sections for more information:

- To create a policy from the command line, see Creating encryption policies with PROUTIL EPOLICY on page 208.
- To create a policy with Data Admin, see Creating encryption policies with Data Admin on page 209.
- To create a policy with OpenEdge SQL, see Encrypting your existing data on page 209.

Creating encryption policies with PROUTIL EPOLICY

PROUTIL EPOLICY MANAGE creates encryption policies for Type I areas and for objects in Type II areas. The basic syntax for creating an encryption policy is:

```
proutil db-name -C epolicy manage object-type encrypt object-name
```

OpenEdge supports the object ciphers described in the following table:

Table 40: Object ciphers

ID	Cipher	Mode	Size	Key type
0	NULL	NULL	—	—
1	AES	CBC	128	binary
2	AES	CBC	192	binary
3	AES	CBC	256	binary
4	DES	CBC	56	binary
5	DES3	CBC	168	binary

The *object-type* is one of the following: `area`, `index`, `lob`, or `table`, and the *object-name* is the name of the object, in quotes if necessary.

For example, the following command creates an encryption policy with the default cipher on a Type I area named `DataArea100` in a database named `t1demo`:

```
proenv>proutil t1demo -C epolicy manage area encrypt "DataArea100"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Encryption policy setting for Area DataArea100 in Area 100
Cipher specification setting to AES_CBC_128 completed.
```

Note that EPOLICY MANAGE operations on areas must be performed while the database is offline. See the PROUTIL EPOLICY MANAGE qualifier on page 1216 for the complete syntax of EPOLICY MANAGE.

Creating encryption policies with Data Admin

You can create encryption policies for objects in your encryption-enabled database using the **Edit Encryption Policy** dialog of **Data Administration**. Objects must reside in a Type II area to be assigned a specific encryption policy. (You can encrypt an entire Type I area with PROUTIL EPOLICY. See Creating encryption policies with PROUTIL EPOLICY on page 208 for details.)

From **Data Administration** or the character **Data Dictionary**, choose **Admin** > **Security** > **Encryption Policies** > **Edit Encryption Policy** to define an encryption policy for a database object. Follow the dialog boxes to select the objects to be encrypted and the desired cipher. For details on the Edit Encryption Policy dialog boxes see the online Help or *OpenEdge Development: Basic Development Tools*.

Your changes are not committed to the database until you click **Commit**. If you do not want to save your changes, clicking **Revert**, cancels the highlighted change. If you do choose to commit your changes, you are asked to confirm. Remember, defining encryption policies does not encrypt data, it signals that all future writes of the data are encrypted.

Encrypting your existing data

There are several different ways to encrypt your existing data. The method you choose depends on the security needs of your database. The methods include:

- Allowing the normal course of database updates to encrypt the data. If an encryption policy has been applied, every time an unencrypted block is updated, the block is encrypted before to being written back to the database. If the data you are encrypting is updated regularly, all the records are encrypted during the normal course of operations.
- Dumping and loading your data. If you can interrupt your normal database operations, dumping and loading the database objects you have enabled for encryption encrypts all the data during the load.
- Encrypt the data with PROUTIL EPOLICY MANAGE UPDATE, as described in the next section.

EPOLICY MANAGE UPDATE EXAMPLE

In the following example, an encryption policy is set for the Type I area `DataArea101`. The normal course of updating records encrypts some of the blocks, and then EPOLICY MANAGE UPDATE completes the encryption. EPOLICY SCAN is used to verify that your data is becoming encrypted, and to check on the progress of the encryption. The general process is as follows:

1. Create an encryption policy for the area:

```
proenv>proutil t1demo -C epolicy manage area encrypt "DataArea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Encryption policy setting for Area DataArea101 in Area 101
Cipher specification setting to AES_CBC_128 completed.
```

2. Check the status of the area with EPOLICY SCAN:

```
proenv>proutil t1demo -C epolicy scan area "DataArea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
AREA     DataArea101 / 101  CURRENT   AES_CBC_128  V:0   1 of 627 blocks encrypted
```

   Observe that at this point, only one block is encrypted.

3. Allow normal database processing to proceed and re-check the progress of encryption with EPOLICY SCAN:

```
proenv>proutil t1demo -C epolicy scan area "DataArea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
AREA     DataArea101 / 101  CURRENT   AES_CBC_128  V:0   200 of 627 blocks encrypted
```

   At this point, 200 of the blocks have been encrypted.

4. Encrypt the remaining blocks in the area with EPOLICY MANAGE UPDATE:

```
proenv>proutil t1demo -C epolicy manage area update "DataArea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
AREA     DataArea101 / 101  CURRENT   AES_CBC_128  V:0   427 of 627 blocks encrypted
```

   MANAGE AREA UPDATE reports encrypting the remaining 427 blocks of the area that were not previously encrypted.

5. Confirm that all the blocks in the area are encrypted with an additional run of EPOLICY SCAN:

```
proenv>proutil t1demo -C epolicy scan area "DataArea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
AREA     DataArea101 / 101  CURRENT   AES_CBC_128  V:0   627 of 627 blocks encrypted
```

   All the data in the area is now encrypted.

All new data is encrypted

If you set an encryption policy on an empty database object, as blocks are written to that object, they are encrypted.

The following steps set an encryption policy on an empty Type I area, and then shows the status of the area after a table is defined and loaded with data:

1. Create an encryption policy for an empty area with EPOLICY MANAGE ENCRYPT:

```
proenv>proutil t1demo -C epolicy manage area encrypt "DataArea100"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Encryption policy setting for Area DataArea100 in Area 100
Cipher specification setting to AES_CBC_128 completed.
```

2. Create a table in the area and load data into it (not shown).
3. Check the encryption status of the area with EPOLICY SCAN:

```
proenv>proutil t1demo -C epolicy scan area "DataArea100"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
AREA     DataArea100 / 100  CURRENT   AES_CBC_128  V:0   127 of 127 blocks
encrypted
```

As new records are written to the `DataArea100` area, they are encrypted. The output shows that the area has grown from 1 block to 127 blocks and that they are all encrypted.

Running with Transparent Data Encryption enabled

Transparent Data Encryption can impact the regular operation of your database as discussed in the following sections:

- Manual start on page 211
- Autostart on page 212
- Temporary files on page 213
- Restricted utility access on page 213
- Database backups on page 214
- Moving tables and indexes on page 214
- Binary dump and load of data on page 217

Manual start

Once you have enabled encryption on your database, every time the database is opened the key store must be authenticated.

If you enable autostart, the authentication is done automatically. If autostart is not enabled, you must manually authenticate. To manually authenticate the key store, every utility, server, or client that opens the database, must supply a passphrase.

ABL clients specify a passphrase for key store authentication by adding the `-KeyStorePassPhrase` argument, followed by the passphrase to the CONNECT statement, as shown:

```
-KeyStorePassPhrase passphrase
```

It is the responsibility of the ABL application to handle prompting for the passphrase before executing the CONNECT statement.

Database utilities, servers, and self-service clients indicate that they require a prompt for key store authentication by adding the following parameter to the command line:

```
-Passphrase
```

You will be prompted for the passphrase, but there is no echoing of keystrokes to the window.

If you have specified the key store user account for autostart, you can override your autostart authentication by adding `-Passphrase` or -KeystorePassPhrase, as appropriate for the command.

If access to the key store cannot be successfully authenticated, the database cannot be opened.

Autostart

Autostart delivers a passphrase you configure to open the key store automatically, so there is no prompting for a key store passphrase. Autostart mode does not impact scripts, but potentially gives unfettered access to encrypted data.

Autostart can be configured when you enable your database for encryption, by adding `-Autostart {admin | user}` to the PROUTIL ENABLEENCRYPTION command. You can change the autostart configuration with the PROUTIL EPOLICY MANAGE command at a later point in time.

Change autostart configuration

To change your autostart configuration to the key store user account:

1. If you did not supply a key store user account passphrase when you enabled your database for encryption, create one now using PROUTIL EPOLICY MANAGE. You must specify the key store admin passphrase with `-Passphrase` if the database is not enabled for autostart with the admin account.

   The command and output is as follows:

```
proenv>proutil t1demo -C epolicy manage keystore userphrase -Passphrase
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Enter the key store passphrase for database t1demo:

This command modifies encryption access control in the Keystore file. After successful
 completion of the command, the Keystore file must be backed-up. (15518)
Enter new passphrase [required] :

Please Retype your Passphrase for Verification
Enter new passphrase [required] :

proenv>
```

2. Change your autostart configuration by specifying key store user account:

The command and output is as follows:

```
proenv>proutil t1demo -C epolicy manage autostart user -Passphrase
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Enter the key store passphrase for database t1demo:

Manage autostart enable has been selected. (15523)
This command modifies encryption access control in the Keystore file.
After successful completion of the command, the Keystore file must be
backed-up. (15518)

proenv>
```

Disabling autostart

You can disable autostart at any time using PROUTIL EPOLICY MANAGE, as shown:

```
proenv>proutil t1demo -C epolicy manage autostart disable -Passphrase
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Enter the key store passphrase for database t1demo:

Manage autostart disable has been selected. (15523)
This command modifies encryption access control in the Keystore file. After
successful completion of the command, the Keystore file must be backed-up.
(15518)
proenv>
```

Observe that `-Passphrase` is added to the command line to override the autostart user account configuration.

Temporary files

The Save Temp Files (`-t`) startup parameter makes OpenEdge temporary files visible. You cannot use this parameter with an encryption-enabled database. If you try to start an encryption-enabled database with `-t`, the following error occurs:

```
Connection to an encrypted database is not permitted with -t. (15389)
```

Restricted utility access

Some database utilities allow access to decrypted data for maintenance and repair. To prevent database utilities from becoming a "back door" for circumventing encryption policies, access to certain utilities is restricted to authenticated database administrators. In general, utilities that manage encryption policies and the key store, and utilities that can provide a clear-text version of encrypted data are restricted. For the following list of utilities, access is denied to all users except an authenticated Database Administrator:

- PROUTIL EPOLICY
- PROUTIL ENABLEENCRYPTION
- PROUTIL DISABLEENCRYPTION
- PROUTIL TABLEMOVE

- PROUTIL IDXMOVE
- PROUTIL TRUNCATE BI
- PROUTIL DUMP (binary dump)
- RFUTIL AIMAGE BEGIN
- DBTOOL

Database backups

When you backup an encrypted database, the backup file contains encrypted blocks. As blocks are written to the backup file, they are encrypted with the current encryption policy. To guarantee that the blocks are backed up with the current encryption policy, there can be no policy changes while the backup is running.

If you restore an encrypted database from backup with `-newinstance`, you will not be able to open the restored database until you rebind the key store with PROUTIL EPOLICY MANAGE KEYSTORE REBIND. For the complete syntax, see PROUTIL EPOLICY MANAGE qualifier on page 705.

Moving tables and indexes

Moving tables and indexes within an encryption-enabled database can cause data to become encrypted or unencrypted. The syntaxes of PROUTIL INDEXMOVE and PROUTIL TABLEMOVE remain unchanged. The table below describes the results of moving objects (tables or indexes) in an encryption-enabled database.

Table 41: Encryption impact of moving tables and indexes

If you are moving...	To...	Then...
A table or index from an unencrypted Type I area	An unencrypted Type I area	• Execution of this operation is open • No change in encryption policy • Object remains unencrypted
	An encrypted Type I area	• Execution of this operation is restricted to an authenticated DBA • No change in encryption policy (the policy for the destination area is applied to the object) • Object is encrypted in the destination area
	A Type II area	• Execution of this operation is open • No change in encryption policy • Object remains unencrypted

If you are moving...	To...	Then...
A table or index from an encrypted Type I area	An unencrypted Type I area	• Execution of this operation is restricted to an authenticated DBA • No change in encryption policy (the source policy is for the entire area) • Object is unencrypted in the destination area • User is prompted to confirm the move and the resulting decryption of the object
	An encrypted Type I area	• Execution of this operation is restricted to an authenticated DBA • No change in encryption policy (the source and target policies are for the entire area) • Object is encrypted in the destination area with encryption policy of the destination area
	A Type II area	• Execution of this operation is restricted to an authenticated DBA • A new encryption policy is created for the object (with the same cipher as the source area policy) • Object is encrypted in the destination area

If you are moving...	To...	Then...
An unencrypted table or index from a Type II area	An unencrypted Type I area	• Execution of this operation is open • No change in encryption policy • Object remains unencrypted
	An encrypted Type I area	• Execution of this operation is restricted to an authenticated DBA • No change in encryption policy (the area policy of the destination area is used) • Object is encrypted in the destination area with encryption policy of the destination area
	A Type II area	• Execution of this operation is open • No change in encryption policy • Object remains unencrypted

If you are moving...	To...	Then...
An encrypted table or index from a Type II area	An unencrypted Type I area	• Execution of this operation is restricted to an authenticated DBA • The encryption policy on the object is removed • Object is unencrypted in the destination area • User is prompted to confirm the move and the decryption of the object
	An encrypted Type I area	• Execution of this operation is restricted to an authenticated DBA • The encryption policy on the object is removed (and the encryption policy of the destination area is applied) • Object is encrypted with the policy of the destination area
	A Type II area	• Execution of this operation is restricted to an authenticated DBA • No change in encryption policy (the object policy remains in use) • Object remains encrypted

Binary dump and load of data

By default, the contents of binary dump and audit archive files are **not** encrypted, even if the database objects being dumped or archived are encrypted. You can optionally add encryption via passphrase-based encryption to your binary dump and audit archive files to increase security of the dump and archive files. As long as your database is enabled for encryption, you can encrypt your binary dump or audit archive, even if the tables you are dumping are not encrypted.

To encrypt the output of binary dump or audit archive, add one of the following case-sensitive qualifiers to the command line:

```
-Cipher 11
```

```
-Cipher 12
```

Note: OpenEdge versions prior to Version 12.0 permitted the use of `-Cipher 6` for an Enterprise database enabled for Transparent Data Encryption. In Version 12.0, if PROUTIL DUMP specifies `-Cipher 6`, the command will continue with a warning:

```
Warning: The minimum allowed cipher for this operation is AES128_CBC_PBE.(19384)
Substituting cipher AES128_CBC_PBE for requested DES_CBC_PBE. (19385)
```

You will be prompted to enter and verify a passphrase. This passphrase is unique to this dump file. If you forget the passphrase, there is no way to recover the contents of the dump file.

When you load your data, you must add the appropriate `-Cipher` qualifer to the command line to indicate that you are loading encrypted data. For dump files created by OpenEdge versions prior to Release 12, add `-Cipher 6` to the command line. For dump files created by OpenEdge Release 12 and later, add `-Cipher 11` to the command line for dump files encrypted using `-Cipher 11`, or `-Cipher 12` to the command line for dump files encrypted using `-Cipher 12`. You will be prompted to enter and verify the passphrase that you entered when you created the dump file.

Note: If when dumping your data, you received a warning as a result of adding `-Cipher 6` to the PROUTIL DUMP command as documented above, you will be required to enter `-Cipher 11` to the PROUTIL LOAD command when loading the dump file.

If you choose to dump or archive encrypted data into an unencrypted dump or archive file, you should load the dump or archive file and delete it as soon as possible to minimize the time the data is stored in an unencrypted format.

Auditing Transparent Data Encryption

To increase the security of your encrypted data, the auditing capabilities of OpenEdge are expanded to record events related to Transparent Data Encryption. The complete set of auditing policies related to Transparent Data Encryption are added to the `policies.xml` file. There is also a single, incremental audit policy in the file, `policies_dbenc.xml`, that can be imported into an existing auditing configuration. For more information on auditing, see *OpenEdge Getting Started: Core Business Services - Security and Auditing.*

The table below lists the audit events related to Transparent Data Encryption.

Table 42: Auditing events for Transparent Data Encryption

Event ID	Event Name	Detects	Operation
11000	`_sys.db.enc.enable`	Enable encryption	PROUTIL ENABLEENCRYPTION
11001	`_sys.db.enc.disable`	Disable encryption	PROUTIL DISABLEENCRYPTION
11100	`_sys.ks.create`	Create a new OpenEdge key store	PROUTIL ENABLEENCRYPTION; PROUTIL EPOLICY MANAGE key store reconstruct
11101	`_sys.ks.delete`	Delete an existing OpenEdge key store	PROUTIL DISABLEENCRYPTION

Event ID	Event Name	Detects	Operation
11102	_sys.ks.open.pass	Successfully opened an OpenEdge key store	PROUTIL EPOLICY MANAGE;internal ABL and SQL commands
11103	_sys.ks.rekey	Changed OpenEdge key store's encryption key	Not in use for this release
11104	_sys.ks.setcipher	Changed OpenEdge key store encryption cipher	Not in use for this release
11105	_sys.ks.setadmin.pwd	Changed OpenEdge key store's admin passphrase	PROUTIL EPOLICY MANAGE key store adminphrase …
11106	_sys.ks.setuser.pwd	Changed OpenEdge key store's user passphrase	PROUTIL EPOLICY MANAGE key store userphrase …
11107	_sys.ks.ke.create.pass	Successfully created new encryption key entry	PROUTIL ENABLEENCRYPTION; PROUTIL EPOLICY MANAGE key store reconstruct
	_sys.ks.ke.update.pass	Successfully changed a key entry's owner, passphrase or state	PROUTIL DISABLEENCRYPTION
11109	_sys.ks.ke.delete.pass	Successfully deleted a key entry	Not in use for this release
11110	_sys.ks.ke.read.pass	Successfully returned a clear-text encryption key	_mprosrv; database utilities; ABL single-user database connection
11111	_sys.ks.open.fail	Failed when attempting to open a OpenEdge key store	_mprosrv; database utilities; ABL single-user database connection
11112	_sys.ks.ke.create.fail	Failed when attempting to create a new key entry	PROUTIL ENABLEENCRYPTION
11113	_sys.ks.ke.update.fail	Failed when attempting to change a key entry's owner, passphrase, or state	PROUTIL DISABLEENCRYPTION
11114	_sys.ks.ke.delete.fail	Failed when attempting to delete a key entry	Not in use for this release
11200	_sys.as.create.pass	Successfully created new autostart credentials (for key store access)	PROUTIL ENABLEENCRYPTION

Event ID	Event Name	Detects	Operation
11201	_sys.as.delete.pass	Successfully deleted existing autostart credentials (for key store access)	PROUTIL DISABLEENCRYPTION
11202	_sys.as.open.pass	Successfully opened autostart credentials and accessed the key store	_mprosrv; database utilities; ABL single-user database connection
11203	_sys.as.recover.pass	Successfully forced new autostart credentials and OpenEdge key store passphrase credentials after lockout condition entered	PROUTIL EPOLICY MANAGE key store reconstruct
11204	_sys.as.update.pass	Successfully updated autostart credentials (for key store access)	PROUTIL EPOLICY MANAGE KEYSTORE [autostart | userphrase | rebind];PROUTIL EPOLICY MANAGE AUTOSTART
11205	_sys.as.open.fail	Successfully opened autostart credentials but failed to access the key store	_mprosrv; database utilities; ABL single-user database connection
11206	_sys.as.update.fail	Failure found when updating autostart credentials	PROUTIL EPOLICY MANAGE KEYSTORE [autostart | userphrase | rebind]PROUTIL EPOLICY MANAGE AUTOSTART
11207	_sys.as.recover.fail	Failed to forced new autostart credentials and OpenEdge key store passphrase credentials after lockout condition entered	PROUTIL EPOLICY MANAGE key store reconstruct
11300	_sys.enc.scan	Started a scan of an encrypted object to determine the # of blocks related to each object security policy	PROUTIL EPOLICY SCAN
11301	_sys.enc.update	Started an update of an encrypted object to re-encrypt previous policy encrypted block with the current policy	PROUTIL EPOLICY UPDATE

Event ID	Event Name	Detects	Operation
11400	_sys.db.dbpolicy.create	Create a new version of a database master key's security policy	PROUTIL ENABLEENCRYPTION
11401	_sys.db.dbpolicy.update	Updated an existing database master key's security policy's information	Not in use for this release
11402	_sys.db.dbpolicy.delete	Deleted an existing database master key's security policy [version]	PROUTIL DISABLEENCRYPTION
11500	_sys.db.objpolicy.create	Create a new version of a database object's security policy	PROUTIL EPOLICY MANAGE;internal ABL and SQL commands
11501	_sys.db.objpolicy.update	Update an existing database object security policy's state	PROUTIL EPOLICY MANAGE;internal ABL and SQL commands
11502	_sys.db.objpolicy.delete	Delete an existing [version] of a database object's encryption policy	Dictionary: delete table/index SQL: DROP table/index … proutil: TBD
11600	_sys.db.pwdpolicy.create	Create a new version of a database passphrase rules policy	PROUTIL ENABLEENCRYPTION
11601	_sys.db.pwdpolicy.update	Update an existing [version] of a database passphrase rules policy	Not in use for this release
11602	_sys.db.pwdpolicy.delete	Delete an existing [version] of a database passphrase rules policy	PROUTIL DISABLEENCRYPTION

Data Admin provides three reports to track audited activities related to transparent data encryption. The reports are as follows:

- **Track Encryption Policy Changes Report** — Reports any events related to encryption policy maintenance. Events with IDs 11400-11402, 11500-11502, and 11600-11602 are tracked by this report.
- **Track Key-store Changes Report** — Reports any events related to the key store and autostart. Events with IDs 11100-11114 and 11200-11207 are tracked by this report.
- **Database Encryption Administration (Utilities) Report** — Reports events such as enabling and disabling encryption for your database, and scan and update utilities. Events with IDs 11000, 11001, 11300, and 11301 are tracked by this report.

Replication support

You can enable Transparent Data Encryption for a database that is enabled for replication, and also enable replication for a database that is encryption-enabled.

For replication and encryption to work together smoothly, the following requirements exist:

- Transparent Data Encryption must be enabled for both the source and target databases.

  The enabling of encryption on the target database is driven by the source database, either by the normal replication process or by your creating or recreating the target database from an encryption-enabled source database.

- You cannot enable before-image encryption on the target database without recreating the target database from the source database.
- Before-image encryption is not automatically enabled when encryption is enabled online for a source database; however, it is automatically enabled when encryption is enabled offline.
- After-image encryption is automatically enabled for a source database when encryption is enabled either online or offline. If after-imaging is active on the target, after-image encryption is automatically enabled on the target by the Replication agent when encryption is enabled.
- The before-image and after-image encryption policies can differ between the source and the target databases. For example, the source can be enabled for before-image encryption, but the target is not required to be enabled.

For details on managing OpenEdge Replication and Transparent Data Encryption together, see *OpenEdge Replication: User Guide*.

Maintaining Transparent Data Encryption

Once you have established your encryption policies and all your data is encrypted, you need to perform some policy maintenance. Periodically, your encryption policies should be updated with a new key. A new key keeps the encryption cipher the same, but provides new input to the cipher algorithm.

Encryption policies are rekeyed in several ways. See one of the following sections for more information:

- To rekey a policy with Data Admin, see Rekey encryption policies with Data Admin on page 222.
- To rekey a policy from the command line, see Rekey encryption policies with PROUTIL EPOLICY on page 224.
- To rekey a policy with OpenEdge SQL, see Encrypting your existing data on page 209.

You can also view the history of an encryption policy. See Viewing encryption policy history in Data Admin on page 224 for instructions.

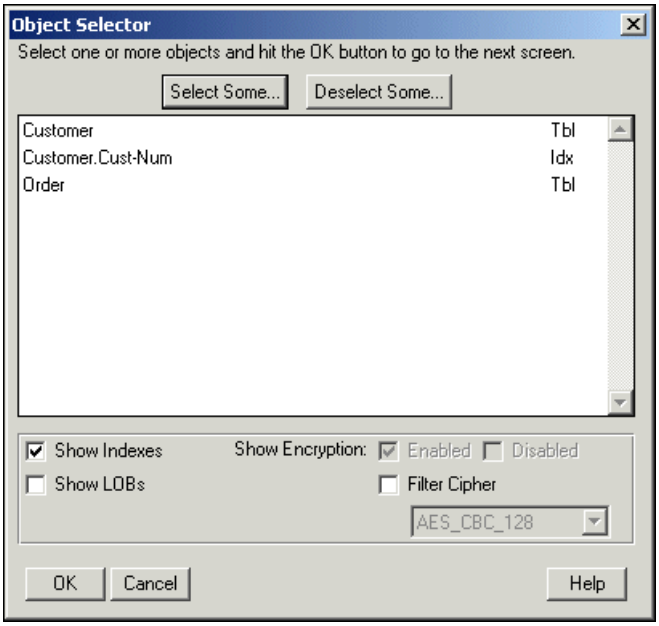
Rekey encryption policies with Data Admin

You can generate new encryption keys for encryption policies in your encryption-enabled database using the **Generate Encryption Keys** dialog. Objects must have an existing encryption policy.

To generate a new encryption key for an existing encryption policy:

1. From **Data Administration**, choose **Admin** > **Security** > **Encryption Policies** > **Generate Encryption Keys**.

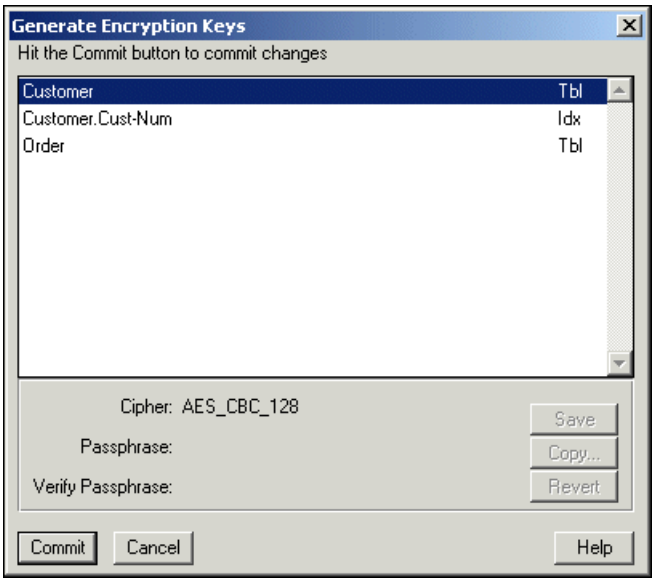
   The **Object Selector**dialog appears:



   By default, only tables and indexes with existing encryption policies are shown. You can also show LOBs with encryption policies by checking **Show LOBs**. You can refine the list by filtering by cipher if you check **Filter Cipher** and choose a cipher.

2. Select the object or objects for which you want to generate new encryption keys and click **OK**.

   The **Generate Encryption Keys** dialog appears:



Note: The **Save**, **Copy**, and **Revert** buttons are disabled, as well as the **Passphrase** and **Verify Passphrase** fill-ins.

3. Click **Commit** to generate new keys for all the objects listed in the **Generate Encryption Keys** dialog, or click **Cancel** to not generate any keys.
4. If you click **Commit**, you are asked to confirm your decision. Click **Yes** or **No**:



Note: Generating new encryption keys is fully supported in the character Data Dictionary.

Rekey encryption policies with PROUTIL EPOLICY

PROUTIL EPOLICY MANAGE allows you to rekey the cipher of an existing encryption policy.

To update the key of a policy, enter the following command:

```
proutil dbname -C epolicy manage object-type rekey object-name
```

Change the cipher of an encryption policy

Similar to rekeying a policy, though a less common task, you can change the cipher of a policy.

To change the cipher of a policy, enter the following command:

```
proutil dbname -C epolicy manage object-type cipher object-name -Cipher cipher-num
```

You can only rekey or change the cipher of objects that have previously been encrypted.

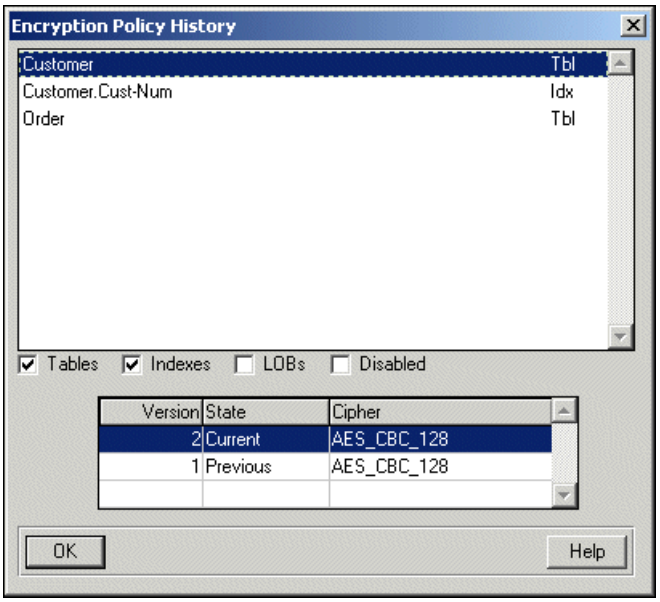
Viewing encryption policy history in Data Admin

You can view the history of an encryption policy in Data Administration. Use the following steps to view encryption policy history for encrypted objects in an encryption-enabled database.

To view encryption policy history:

1. From Data Administration, select **Admin** > **Security** > **Encryption Policies** > **Encryption Policy History**.

   The **Encryption Policy History** dialog appears.



The **Version** of a policy is an integer that starts at zero, and increases every time a policy is modified. The **State** indicates whether the policy is the Current or Previous policy. The **Cipher** indicates the cipher of the policy.

2. In the **Encryption Policy History** dialog, select an object to view its encryption policy history. You can modify the listed objects by checking and unchecking the boxes for **Tables**, **Indexes**, **LOBs**, and **Disabled**.
3. When you finish reviewing encryption policy histories, click **OK** to exit the **Encryption Policy History** dialog.

Disabling encryption

If you determine that you no longer need your database encrypted, you can disable encryption with the PROUTIL DISABLEENCRYPTION command. The basic syntax is:

```
proutil dbname -C disableencryption
```

Disabling encryption decrypts all the data in your database, removes all the encryption policies, and archives your key store (by renaming it to *dbname*`.ksbk`).

If you have a large number of encrypted database objects, you might want to control the decryption of your data before running the DISABLEENCRYPTION command. To decrypt your data, update the encryption policy to the NULL cipher.

To decrypt data:

1. Verify that the object is encrypted with EPOLICY SCAN, as shown:

```
proenv>proutil t1demo -C epolicy scan area "dataarea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
AREA     dataarea101 / 101  CURRENT   AES_CBC_128  V:1   627 of 627 blocks
encrypted
```

2. Change the encryption policy cipher to the NULL cipher (cipher 0), as shown:

```
proenv>proutil t1demo -C epolicy manage area cipher "dataarea101" -Cipher 0
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Encryption policy cipher change for Area dataarea101 in Area 101
Cipher specification change to NULL_NULL_NULL completed.
```

3. Re-confirm the policy change with EPOLICY SCAN, as shown:

```
proenv>proutil t1demo -C epolicy scan area "dataarea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
AREA     dataarea101 / 101  CURRENT   NULL_NULL_NULL  V:2   1 of 627 blocks
encrypted
AREA     dataarea101 / 101  PREVIOUS  AES_CBC_128  V:1   626 of 627 blocks
encrypted
```

4. Update all the blocks to the NULL cipher policy with EPOLICY MANAGE UPDATE, as shown:

```
proenv>proutil t1demo -C epolicy manage area update "dataarea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
```

5. Confirm that the object is decrypted and the policy deleted with EPOLICY SCAN, as shown:

```
proenv>proutil t1demo -C epolicy scan area "dataarea101"
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
No encryption policy exists for object dataarea101.
EPolicy: Encryption policy management failed -402
```

6. Disable encryption, as shown:

```
proenv>proutil t1demo -C disableencryption
OpenEdge Release 10.2B1P as of Fri Nov 20 19:01:52 EST 2009
Updating blocks of object 9/7. (15466)
Updating blocks of object 9/100. (15466)
BI Encryption has been disabled for database t1demo. (12490)
Encryption Feature has been disabled for database t1demo. (12490)
The BI file is being automatically truncated. (1526)
```

By decrypting your data before disabling encryption, you can control the impact of the update to normal database processing.

OpenEdge SQL support for Transparent Data Encryption

OpenEdge SQL Data Definition Language supports Transparent Data Encryption. The new functionality provides **only** the DBA or security administrator with the ability to:

- Define encryption specifications for new tables, indexes, and `LOB` columns
- Modify encryption or decryption specifications for existing tables, indexes, and `LOB` columns
- Drop existing encrypted tables, indexes, and `LOB` columns
- View encryption specifications for tables, indexes, and `LOB` columns
- Rekey existing encrypted tables, indexes, and `LOB` columns

Note that the tables, indexes, and `LOB` columns must be in a Type II storage area. For details on the modified SQL syntax, see *OpenEdge Data Management: SQL Development* and *OpenEdge Data Management: SQL Reference*.

11

Replicating Data

Data replication is the distribution of copies of information to one or more sites. In a single enterprise, sites spanning organizational and regional boundaries often require the timely sharing of transaction data across databases in a consistent manner. Developing and deploying a successful replication process involves careful planning and input from business experts, application developers, and administrators.

For details, see the following topics:

- Replication schemes
- Database ownership models
- Implementing log-based site replication

Replication schemes

A replication scheme is a system of definable tasks and operations that are used to perform data replication. An enterprise's replication scheme addresses its specific business requirements. This section summarizes different replication models, data ownership models, and implementation strategies that are available. A replication scheme can be implemented through event triggers or through log-based capture.

Trigger-based replication

To implement trigger-based replication, use event triggers stored in the database. When an event to be replicated occurs (that is, a record is created, modified, or deleted) the database uses the event to record the change in a replication change log. ABL (Advanced Business Language) provides full support for trigger-based replication. See *Develop ABL Applications* for more information about trigger-based replication.

Log-based site replication

Log-based site implementation (or site replication) is based on a monitor that watches a log of transactions and propagates these changes to other sites. Generated by the database, this log comprises a stream of database transactions. This is a very efficient way of replicating data from server to server. It also allows you to maintain a remote backup of a primary database. The OpenEdge RDBMS supports site replication using after-imaging (AI) files. For more information on log-based site replication see Implementing log-based site replication on page 232.

Asynchronous replication

As of OpenEdge Release 12.0, Progress OpenEdge databases support only the *asynchronous* model of replication.

Asynchronous replication (also known as store and forward replication) allows the replication to occur outside the scope of the original transaction. The replication might take place seconds, minutes, hours, or days from the time of the transaction, depending on your business requirements. Although the replication executes record by record, replication can occur by transaction. That is, if an order is placed in the system with order lines containing multiple data changes and these changes are made within the scope of a single transaction, the changes can be replicated as a single transaction.

Database ownership models

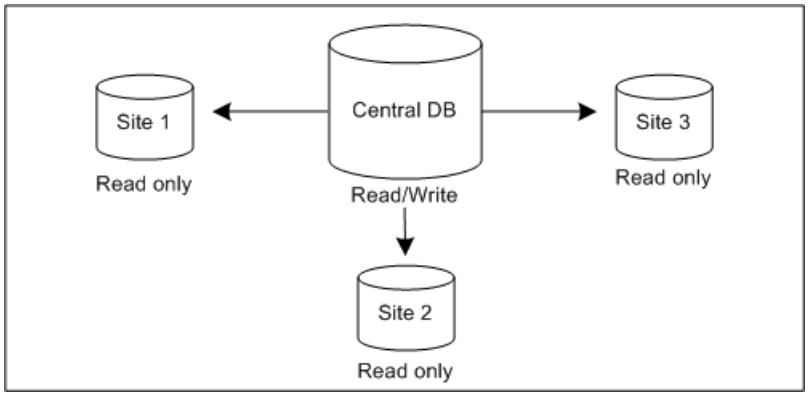
Data ownership models determine how changes to a database (or site) affect other databases in the network. This section describes three models, *data distribution*, *data consolidation*, and the *peer-to-peer* model, and how they relate to replication.

Distribution model

In the distribution ownership model, a single master database owns the data. The master database is the read/write area, and all changes are made to this database only. All changes are then propagated to the remote sites in a read-only state. The remote sites cannot change the data, only view it. In terms of replication, the chief advantage to this model is that it greatly reduces data collision (conflicts between updates to the same record). This is because data changes are made at one site only.

The following figure illustrates the data distribution model.

Figure 19: Data distribution model



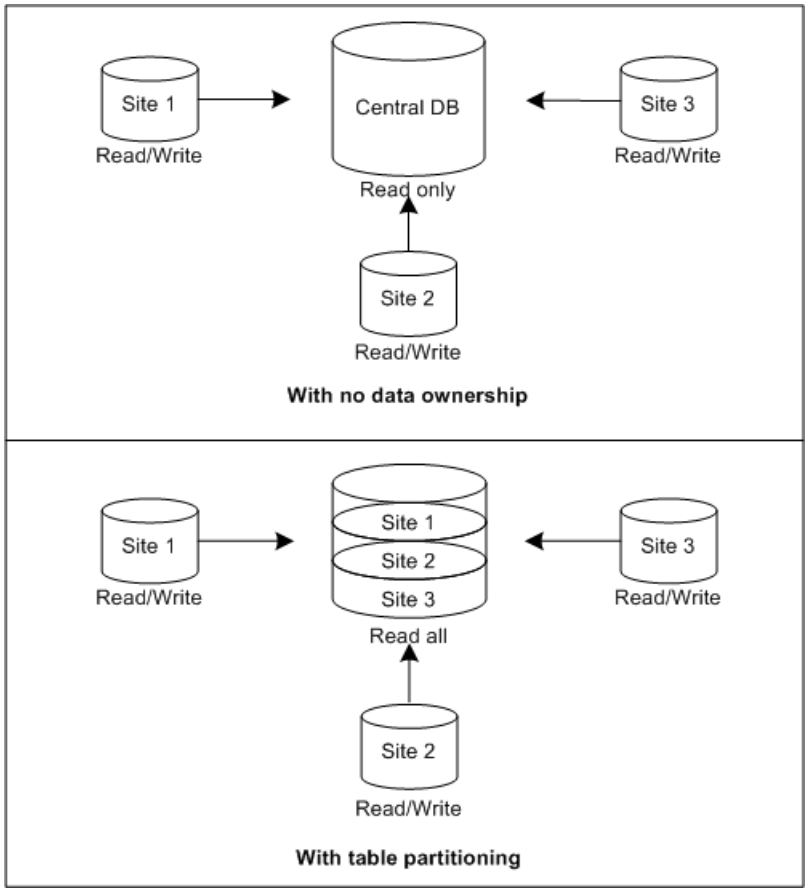
Consolidation model

In the consolidation model, data changes are made at the remote sites and then propagated to the central database. The central database is read-only and is used for reporting purposes. For replication, this model increases the frequency of data collision over the distribution model. If there is a collision of changes by two or more users, the changes are applied on a first-come-first-served basis.

To avoid data collision, the consolidation model often uses *table partitioning.* Table partitioning (also called data ownership) requires that all data be owned by each site. Changes to data at each remote site are made exclusively by respective remote site users. A data ownership model might not be appropriate for your business organization. Although data collisions are avoided, the ability to update the same record from any site is lost.

The following figure illustrates two data consolidation models, one with no data ownership, and the other with table partitioning.

Figure 20: Data consolidation models

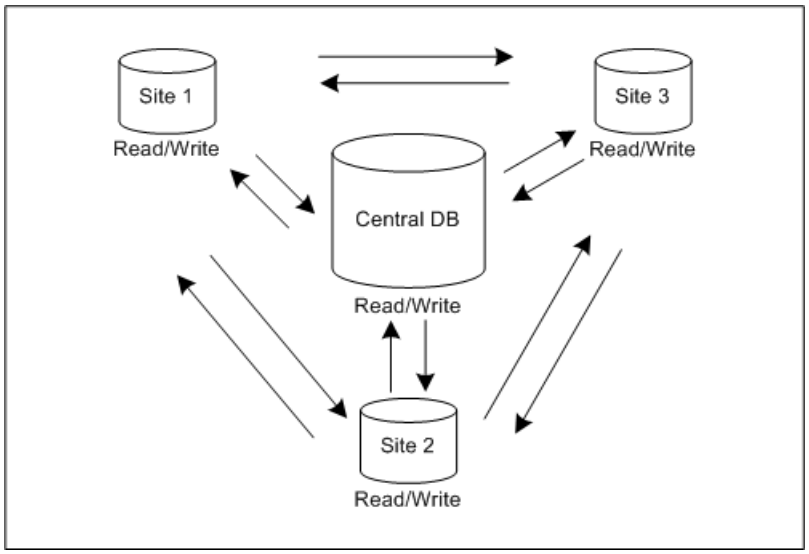


Peer-to-peer model

In a peer-to-peer model (or "update anywhere") any user at any site can update data. This is the most flexible replication model. However, in a peer-to-peer scheme, data collision is a side effect that must be addressed. Data collision must be resolved based on business requirements.

The following figure illustrates the peer-to-peer model.

Figure 21: Peer-to-peer model



Implementing log-based site replication

With log-based site replication, a central database is replicated in its entirety to one or more secondary sites. Log-based site replication allows the following:

- The creation of "hot" standby sites in case the primary site fails
- Flexible snapshot capability to control the timeliness of replication
- A transparent method of maintaining a remote backup of a primary database

Log-based replication procedure

Site replication with OpenEdge databases is implemented through the use of after-imaging and database backup procedures. For complete information about these procedures, see After-imaging on page 139 and Backing Up a Database on page 97.

To implement log-based site replications:

1. Add after-imaging extents and then enable after-imaging in the primary database. For information about after-imaging extents and enabling after imaging, see After-imaging on page 139.
2. Use the PROSTRCT utility with the LIST option to create a structure description file containing the central database's data structure. For information about the structure description file and the PROSTRCT utility, see Maintaining Database Structure on page 325.
3. With the structure description file produced from the central database, use PROSTRCT with the CREATE option to create an additional database on the remote system.
4. Perform a backup of the primary database to initialize the secondary database. This step creates a basis for subsequent roll-forward operations. For information about performing a backup, see Backing Up a Database on page 97.
5. Restore the backup copy of the primary database to the secondary database.

6. Use the RFUTIL command with the option EXTENT FULL to monitor the after-image extents. This will automatically determine which image extent is ready for replication (or transfer) to the secondary site. You can transfer the after-image extent file to the secondary site using an OS command to remote copy.

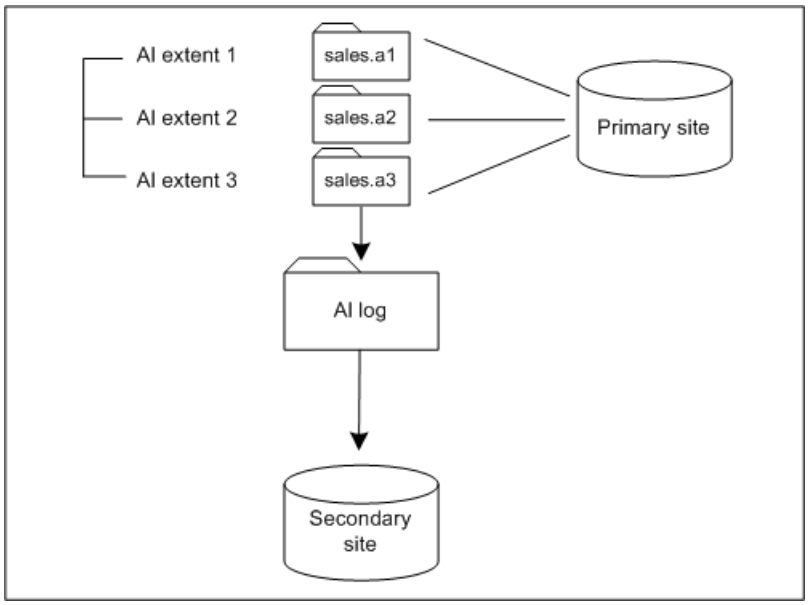
   For more information about RFUTIL, see RFUTIL Utility on page 797.

7. Once the after-image extent has been transferred to the secondary site, use RFUTIL with the EMPTY option to mark the extent "empty" and ready for use on the primary database.
8. Implement a process to monitor and transfer full after-image extents (AI extents). You can copy AI extents to an AI log, then transfer the log contents to the secondary site on a continuous basis.
9. If it becomes necessary to shift operations to the secondary site, transfer the last "full" and "busy" after-image extents and roll-forward to completion. Start up of the secondary site database causes the database to undergo crash recovery, resulting in the shift to the secondary site.

   For more information about performing roll-forward recovery, see Recovering a Database on page 117.

The following figure shows after-image extent files replicated from the primary to a secondary database.

Figure 22: Example of site replication



12

Failover Clusters

Failover Clusters provide an operating system and hardware vendor-independent solution for automated fail over of your OpenEdge database and related resources.

Throughout the remainder of this chapter, Failover Clusters is also referred to as Clusters.

For details, see the following topics:

- Overview
- Security
- Performance
- Logging
- Required cluster components
- Network considerations
- Terms and concepts
- Using the PROCLUSTER command-line interface
- Results of enabling an OpenEdge database for fail over
- Platform-specific considerations
- Using a cluster-enabled database with the AdminServer
- Using a cluster-enabled database with standard commands
- Using the Windows Cluster Administrator
- Emergency disabling of a cluster-enabled database

- UNIX cluster management commands

Overview

A system cluster is comprised of two or more machines, known as nodes, tightly integrated through hardware and software to function together as one machine. In a cluster, redundant hardware and software are primarily put in place to enable fail over. *Fail over* is the movement of a cluster resource from one node in the cluster to another node. If something goes wrong on one node, or the node needs to be taken offline for maintenance, cluster resources can fail over to another node to provide continual access. Disk access is a key factor in proper fail over. If you use a clustered environment and wish to use Failover Clusters, you must have your database on a shared device. A *shared device* is a disk that is available to any node in the cluster. If a node in the cluster has an outage, the shared device is still recognized and available to the remaining nodes in the cluster, thus providing access to the database.

Failover Clusters provides a simple command-line interface, PROCLUSTER, to your operating system's clustering software. The Clusters interface is easy to use, and is the same regardless of the hardware or software platform. This simplifies administration of an OpenEdge database in a clustered environment. Because Clusters integrates your database into your cluster manager software, cluster resource administration and fail over mechanisms are enabled for your database.

Clusters does not replace OS-specific cluster management software. In fact, Clusters requires that the OS cluster management software and hardware be properly configured. See Related software and hardware on page 236 for your specific OS software and hardware requirements.

The operating system integrates specific components of its clustering technology to monitor the system resources' state and fail over the resource to another node if the primary node is not accessible. Clusters tightly integrates with the cluster management software so that OpenEdge is properly defined as a cluster resource and fails over during planned or unplanned outages. A planned outage might be a hardware or software upgrade. An unplanned outage might be a system crash. With Clusters, you can decide ahead of time how clustered resources will behave during fail over. Failover Clusters eliminates unnecessary downtime and provides continuity of behavior in the cluster even when the database administrator managing the cluster is not the one who set it up.

Clusters relies on the operating system cluster management for node selection and ordering. The node the cluster chooses to fail over to is a default preference Clusters accepts from the cluster manager. If you want to tailor this, you must use your operating system cluster management utilities. Clusters does not alter node ordering. All clustered systems provide a means for disallowing hosts from hosting a resource. Clusters does not provide a means to disallow a host; you must use your operating system cluster management utilities to do this.

Related software and hardware

Each operating system vendor defines the hardware requirements for their clustering technology. Progress Software Corporation recommends that customers purchase cluster packages from their operating system vendors according to their operational needs. The following sections describe the minimum clustering requirements and supported versions.

AIX (64-bit)

IBM software:

- AIX V6.1
- HACMP 5.3 cluster manager

HPUX (64-bit PA-RISC and Itanium)

HP software:

- HPUX 11.i v3 or later
- HP mc/ServiceGuard 11.x or later

SUN Solaris Sparc (64-bit)

SUN software:

- SUN Solaris Sparc 10
- SUN Cluster Version 3.1

Windows

Microsoft software:

- WinServer 2003 Enterprise

  You must have WinServer 2003 Enterprise installed on hardware configurations approved and supported by Microsoft. To use the Clusters product, the cluster should have (at a minimum) the following resources configured:

  - A shared physical disk
  - An IP address
  - A cluster name

Installation

Prior to enabling an OpenEdge database for use in a cluster, OpenEdge must be installed with the Enterprise RDBMS license on every node in the cluster; and the installation path must be identical on every node. When upgrading from earlier releases, all cluster-enabled databases must be disabled before being updated and re-enabled afterwards.

You must also ensure there are no active instances of the Cluster Administrator utility prior to installation. To do this in Windows, use the Windows Task Manager to check for processes running the `cluadmin.exe` image. If a user associated with an instance of the Cluster Administrator is not readily identifiable, these processes can be forcefully terminated as follows:

Failure to shut down all instances of the Cluster Administrator tool results in a partial installation and fails to properly register the resource and administrator extension DLLs, causing cluster management to fail.

To terminate instances of the Cluster Administrator utility:

1. Within Task Manager, select the cluadmin.exe process and press the **End Process** button.
2. Answer **yes** to the popup dialog asking if you really want to terminate the process.

Configuration

You must define and modify the environment variables described for your operating system in the following sections for Failover Clusters to operate correctly.

AIX

- Set PSC_CLUSTER_PATH to the install directory for OpenEdge
- Set JREHOME to the Java run-time environment install directory
- Modify PATH to include $PSC_CLUSTER_PATH/bin
- Modify LIBPATH to include $PSC_CLUSTER_PATH/lib, $PSC_CLUSTER_PATH/bin, $JREHOME/lib, $JREHOME/jre/bin, and $JREHOME/jre/classic
- Set DLC to the install directory for OpenEdge

HPUX

- Set PSC_CLUSTER_PATH to the install directory for OpenEdge
- Modify PATH to include $PSC_CLUSTER_PATH/bin
- Modify SHLIB_PATH to include $PSC_CLUSTER_PATH/lib and $PSC_CLUSTER_PATH/bin
- Set DLC to the install directory for OpenEdge

Modifying the java_env script

To modify the $DLC/bin/java_env script:

1. Make a backup copy of the script.
2. Open the script in the editor of choice and locate the string "OSF1".
3. Modify the JDKHOME and JREHOME environment variable values to point to the install directory for each Java package.
4. Save the changes.

SUN Solaris Sparc

- Set PSC_CLUSTER_PATH to the install directory for OpenEdge
- Modify PATH to include $PSC_CLUSTER_PATH/bin
- Modify LD_IBRARY_PATH to include $PSC_CLUSTER_PATH/lib and $PSC_CLUSTER_PATH/bin
- Set DLC to the install directory for OpenEdge

Windows

- Set PSC_CLUSTER_PATH to the install directory for OpenEdge
- DLC/bin/procluster.bak

  If the installation of additional products overwrites the procluster.bat file, the procluster.bak file is saved. If procluster.bat is overwritten, the following error message appears:

```
'pscluster' is not recognized as an internal or external command, operable
program or batch file
```

To correct this problem, copy `procluster.bak` to `procluster.bat`.

Security

OpenEdge conforms to the security model defined by the OS vendor in terms of what users can create and modify, access rights to the various directories and devices, and rights to start and stop resources, such as databases.

Performance

Performance of the database should not be affected by the use of Clusters beyond the additional separate process required to probe and report on the database's viability to the cluster management software. Fail over times depend on the cluster environment and will vary.

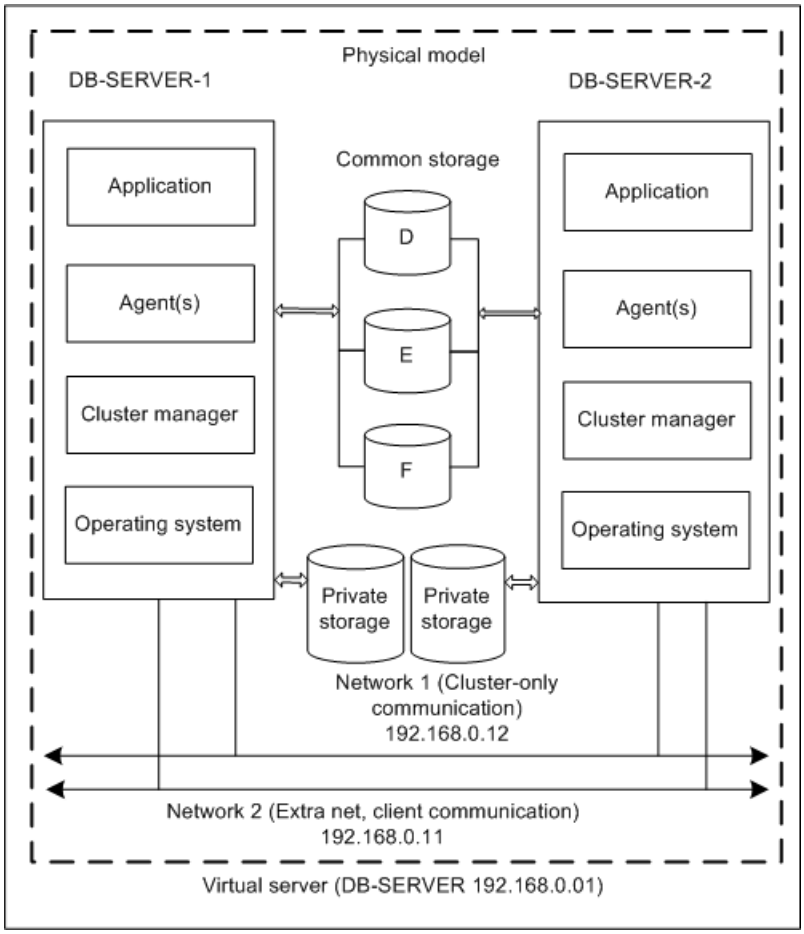
Logging

PROCLUSTER generates a log file, `$PSC_CLUSTER_PATH/PSCluster.log` on UNIX and `%PSC_CLUSTER_PATH%\PSCluster.log` in Windows, that records the creation and management of database resources. The log file tracks the PROCLUSTER commands. The rate of growth for the log file is very small, and it can be deleted at the customer's discretion.

Required cluster components

The following figure shows the components of a typical cluster. For Failover Clusters, OpenEdge requires at least a two-host configuration utilizing a common storage architecture such as SCSI, and systems with redundant network and cluster interconnections. Your database must reside on one or more shared devices, also known as common storage.

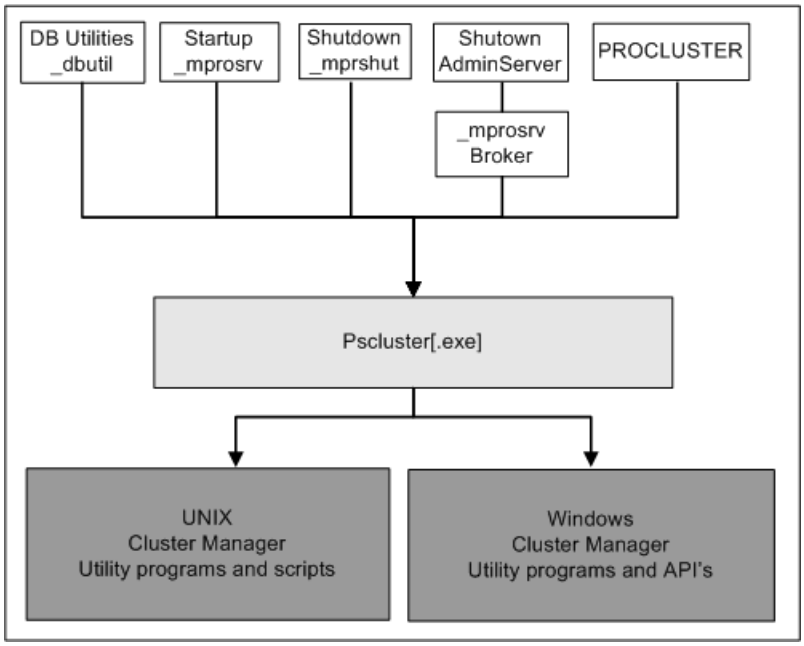
Figure 23: Cluster component configuration



If the database structure changes or the database is moved to a new common storage device, the Clusters command-line interface, PROCLUSTER, makes it very easy to ensure that the database is still protected if a fail over event occurs. If the shared device in the cluster needs to change, use PROCLUSTER to stop the database and enable it on the new shared device once it is moved. If extents are added, Clusters provides for making this change, and you only need to start the database once the changes are automatically applied.

Clusters integrates OpenEdge into the operating system cluster not only by making use of the pre-existing cluster manager software, but by also augmenting OpenEdge feature functionality. When you enable a database for failover, the master block is updated to reflect this fact. When you have a cluster-enabled or cluster-protected database, commands are then funneled through the underlying `pscluster` executable to the operating system cluster manager. The following figure shows this relationship. The cluster manager software must know about OpenEdge to handle it properly in the event of a failover. See Using the PROCLUSTER command-line interface on page 243 for information on using PROCLUSTER to cluster-enable a database.

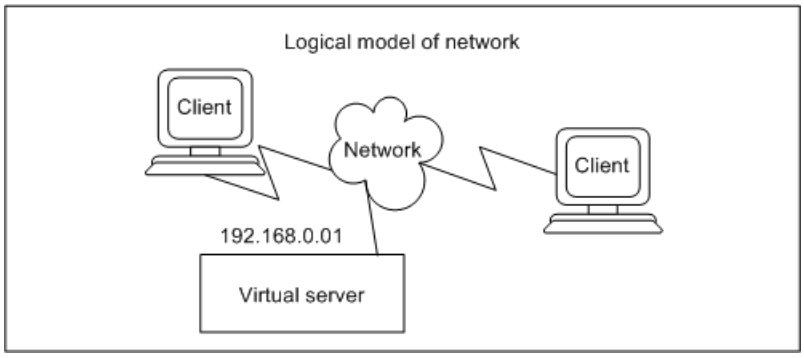
Figure 24: Relationship between database commands and utilities and the cluster manager



Network considerations

Clusters assumes a virtual server model. Connections to the database must be through the cluster alias and cluster IP address, which are different from a single node name and IP address. Figure 23: Cluster component configuration on page 240 shows the physical layout of a cluster where the cluster IP address is 192.168.0.01 (the virtual server IP address). For Clusters, clients connect to this IP address and not the address of one of the individual nodes. In essence, the clients really only know about the virtual server as the node to connect to. The following figure shows the clients connecting over the network to the virtual server with its IP address of 192.168.0.01. Within this cluster there might be several nodes with separate IP addresses, but the clients need not know about each node and must not connect to the individual nodes or Clusters will not work properly.

Figure 25: Virtual network cluster components



Terms and concepts

Clusters is able to offer ease-of-use and maintenance by automating many of the manual procedures you would otherwise need to perform to ensure proper fail over. This section describes the terms and concepts Clusters uses to make this automation possible.

Resources and dependencies

A cluster *resource* is the database and all of its dependencies. A *dependency* is an additional resource that the database requires to successfully function. Dependencies for the database resource include page writers, the after-image writer, the before-image writer, the physical disks the database resides on, the network, etc. A resource must be a member of the cluster. PROCLUSTER registers resources that need to be persisted in a cluster environment. You must register your database as a resource; however, you do not need to identify any dependencies for the database.

Failure and recovery action

The complexity of cluster operation mandates the need for default and automated recovery actions with respect to a group of resources for OpenEdge. A *resource group* is comprised of a resource and all of its dependencies. To determine how a group of resources should behave in a fail over condition, each resource has fail over policies.

A failover *policy* predefines the recovery action that the resource, or a resource group, should automatically follow when a fail over condition occurs.

Fail over policies

The fail over policies for Clusters are as follows:

- **IsAliveInterval** — The polling interval in between calls to the IsAlive method to determine if the application is still alive. The IsAliveInterval is 60 seconds.
- **LooksAliveInterval** — The polling interval in between calls to the LooksAlive method to determine if an application appears alive. The LooksAliveInterval is five seconds.
- **CheckInterval** — The amount of time to attempt to restart a database on its current node. The CheckInterval period is 900 seconds.
- **FailureReportThreshold** — The number of unsuccessful restarts to allow during a CheckInterval period before signaling a fail over request. The FailureReportThreshold is three.
- **Retries** — The number of attempts to start the application before initiating failover. Retries is three.
- **Delay** — The number of seconds to wait prior to initiating fail over. Delay allows the operating system time to perform clean up or other transfer related actions. Delay is 0 seconds.
- **FailBack** — Indicates if the application is to be transferred back to the primary node upon its return to operation. FailBack is False (Windows).
- **AutoStart** — Indicates to the cluster software that the application should automatically start without operator intervention. AutoStart is True (Windows).

Using the PROCLUSTER command-line interface

The PROCLUSTER command-line interface provides a user-friendly interface to Clusters. For the complete PROCLUSTER syntax, see PROCLUSTER utility on page 836.

Cluster-enabling a database

Once a database is created and resides on a shared disk, you can enable the database as a cluster resource so that it will fail over properly. The database must not be in use when it is enabled as a cluster resource. Enable the database as a cluster resource with the following command:

```
procluster db-name enable [-pf params-file] [AI] [BI] [APW=n] [WDOG]
```

You must specify the fully qualified path of the database you want to enable for fail over. The database must be located on a shared disk among the cluster nodes, and the shared disk must be online for the current node. The parameter file contains any parameters that the database requires when started. The parameter file is required to:

- Be named *db-name*`.pf`
- Reside in the same directory as the database `.db` file
- Contain the parameter `-cluster protected`
- For Solaris only, the length of the mount point of the database cannot exceed 18 characters

If the parameter files does not meet these requirements, database startup will fail.

When you enable a database as a resource using PROCLUSTER enable, Clusters performs the following:

- Examines the database structure and generates a registration file
- Logs the registration in the database's log file
- Performs the cluster-specific registration

A cluster-enabled database:

- Cannot be started without following the prescribed protocol—see Starting a cluster-enabled database on page 245.
- Cannot be deleted without first being disabled—see Disabling a cluster-enabled database on page 244.
- Cannot have its physical structure altered without being re-registered—see Changing the structure of the database on page 247.

If PROCLUSTER enable is successful, it returns the following message:

```
The cluster REGISTER command was successful. (10532)
```

To verify the success of PROCLUSTER enable, examine the database log file (`.lg`) to see if the following messages occur:

```
xx.xx.xx prostrct cluster session begin for userid on CON:. (451)
xx.xx.xx prostrct cluster session end. (334)
```

To verify that the database is created and available on the machine as a cluster resource in Windows, you can use the Cluster Administrator to verify that the fully qualified path of the database is visible within the Virtual Server Group area of the tool. On UNIX, you can use the operating system-specific command to enumerate cluster objects. In the list of enumerated objects, the database with a UUID appended to the end will be listed, with the target and state displayed as offline, as shown in the following example:

```
db_nameE3B35CEF-0000-0000-DC17-ABAD00000000
```

The text is the UUID for the newly created database resource. For more information on cluster administration tools and commands, see your operating system cluster documentation.

Note: The PROCLUSTER ENABLE will register the database as a cluster resource, even if there are errors in the command for the helper processes. To correct the errors, you must first use PROCLUSTER DISABLE to unregister the database, then use PROCLUSTER ENABLE to re-register without errors.

Disabling a cluster-enabled database

If a database is already enabled and you want to remove the database as a cluster resource, enter the following command:

```
procluster db-name disable
```

You must specify the fully qualified path of the database to disable. Specifying the database name automatically disables any other optional dependencies specified when the database was enabled.

When you remove a database resource using PROCLUSTER disable, Clusters does the following:

- Shuts down the database if it is running
- Deletes the resource from the cluster manager software once it is in an offline state
- Deletes the group from the cluster manager software if the resource is the last resource in the resource group

You can use the Cluster Administrator tool in Windows or you can use the operating system specific command to enumerate cluster objects on UNIX, to ensure that the database has been removed as an enabled cluster resource. For more information on cluster administration tools and commands, see your operating system cluster documentation.

Starting a cluster-enabled database

To start a cluster-enabled database, enter the following command:

```
procluster db-name start
```

The database to be started is specified by *db-name*. The database must have been previously enabled as a cluster resource. The database name must contain the fully qualified path.

Note: PROCLUSTER will append `-pf db-name.pf` to the `proserve` command that is generated to start the database. The start command will fail if this parameter file is not found. If any of the helper (APW, BIW, or AIW) processes or the watchdog (WDOG) process specified to start with a particular database fails, they will not be started on another machine during a failover. They produce an OFFLINE status.

To verify that the database started correctly, use the `isalive` and `looksalive` parameters to PROCLUSTER. See Isalive and looksalive on page 393 for more information.

Stopping a cluster-enabled database

To stop a cluster-enabled database using normal shutdown procedures, enter the following command:

```
procluster db-name stop
```

The database to be stopped with Clusters is specified by *db-name*. The database must be a member of the cluster. The database name must contain the fully qualified path. To verify that the database has successfully stopped, use the `Isalive` and `Looksalive` parameters of PROCLUSTER. For more information, see Isalive and looksalive on page 393.

When you stop the database with PROCLUSTER stop, Clusters does the following:

- Stops the database
- Notifies the cluster that the resource should be stopped without fail over

---

Note: PROCLUSTER stop can be used interchangeably with PROSHUT or the AdminServer to stop a database.

Terminating a cluster-enabled database

To force a registered or cluster-enabled database to shut down, enter the following command:

```
procluster db-name terminate
```

The database to be forcefully shut down with Clusters is specified by *db-name*. The database must be a member of the cluster. The database name must contain the fully qualified path.

Isalive and looksalive

The `isalive` and `looksalive` parameters query the cluster manager software about the state of a database. With Isalive, the query is made to the active system and the check is known to be accurate. With Looksalive, on systems where the cluster manager maintains a cache of the cluster state, PROCLUSTER will query the cache instead of the active system. There is a small window of opportunity where the cache and the system could be out of sync.

To determine if a resource looks operational, enter the following command:

```
procluster db-name looksalive
```

The looksalive exit status returns the following text if the database looks alive:

```
Resource: db-name State: Looks Alive
```

The value of Looks Alive is only returned when the database is enabled and started. All other states return the following, where *db-name* is the fully qualified path of the database:

```
Resource: db-name State: Not Alive
```

To determine if a resource is actually operational, enter the following command:

```
procluster db-name isalive
```

The isalive exit status returns the following text if the database returns a successful query:

```
Resource: db-name State: Is Alive
```

All other states return the following:

```
Resource: db-name State: Not Alive
```

Results of enabling an OpenEdge database for fail over

Enabling OpenEdge as a cluster resource creates the file *<dbname>*`Resources.cluster`. For example, if the database name is `sports2000.db`, the cluster resource filename will be `sports2000Resources.cluster`. This is a text file that contains names of resources dependent on the database resource. If the database was not enabled with any helper processes or performance enhancers like AIW, APW, BIW, WDOG, the `.cluster` file will be empty; otherwise, it will contain the resource names of the dependent resources as registered with the cluster manager. Do NOT edit or delete this file. Failover Clusters manages this file, and deletes it when the database is disabled.

Database UUID file (HPUX 64 bit only)

In addition to the `.cluster` file, on HPUX, a UUID file is also created. This file contains a translation of the process names to the cluster-registered names. The file is named *<dbname>*`.uuid`. For example, if the database name is `sports2000.db`, the UUID file will be `sports2000.uuid`.

Changing the structure of the database

When you modify the structure of a cluster-enabled database using the PROSTRCT utility, Clusters does the following:

- Automatically identifies that the resource needs enabling
- Takes the resource offline if it is running
- Disables and then re-enables the resource automatically

PROSTRCT handles the updating of the cluster resource list of the cluster manager automatically. For more information on PROSTRCT, see PROSTRCT utility on page 786

After the change is complete, you must manually start the database with PROCLUSTER.

Adding extents on a volume group or file system different from the database (AIX only)

Additional steps are required on AIX to add extents to a database on a volume group or file system that is different from the current location of the database. After modifying the extents with PROSTRCT, a file named `PSC_CLUSTER_REG.TMP` is created in the `scripts` directory located in your install directory. This file must be reviewed and executed.

Review the `PSC_CLUSTER_REG.TMP` file to make sure that it contains the correct path of all the added extents. This information is towards the end of the file with the title DatabaseDependency. If the paths are incorrect, see the following the procedure.

Note: Do not add extents to a volume group or file system already in use by another resource group.

To correct and register the new extents with the cluster manager:

1. Edit `PSC_CLUSTER_REG.TMP` and correct the paths for all the extents you added.
2. Change your directory to `$PSC_CLUSTER_PATH`.
3. Run the following command, which might take a few minutes to complete:`bin/pscluster register scripts/PSC_CLUSTER_REG.TMP`

Platform-specific considerations

There are small differences in the implementation of Failover Clusters on the various operating systems where it is supported. The following sections note important differences.

Adding nodes where the database can be run for AIX

By default, the resource group is enabled to run ONLY on the current node, the node that was used to enable the database. You change the priority and the number of nodes the database can run on through the **SMIT** menu, using the `smit hacmp` fast path command. From this menu, choose **Cluster Configuration** > **Cluster Resources** > **Define Resource Group** > **Change/Show a resource group**. The **Change/Show a resource group** provides a menu where you can change node selection.

After adding the nodes to the resource group, ensure that the directory and its contents for the resource group are copied to the other nodes where the database is enabled to run. The directory of the resource group is located under `/etc/cluster`. The name of the directory will be the name of the resource group. The directory and its contents must be in the same location on all nodes.

Upper limit on the number of packages for HPUX

There is an upper limit on the number of packages that can be registered with the cluster manager. One package will be associated with the database. The Watch Dog, BIW, AIW, and every APW will each be associated with an additional package. For example, if the database is enabled as a cluster resource along with an AIW, a BIW, a WDOG, and three APWs, a total of seven resources will be registered with the cluster manager. PROCLUSTER allows for the creation of files and directories necessary for registration even if the limit is exceeded. It is important to review your configuration details and verify that the maximum number of packages for the cluster will not be exceeded prior to enabling the database as a cluster resource.

If you exceed the limit, adjust the MAX PACKAGE LIMIT. You can then fix your database resource using one of the following methods:

- Use the `cmapplyconf` utility provided by MC/Service Guard to register the packages manually.

  Note: The package names can be found in *`dbname`*`.uuid` file.

- Disable the database and then re-enable with the desired packages. PROCLUSTER will ensure automated registration.

Directory of registered packages for HPUX

Under the directory `/etc/cmcluster`, you will find a directory for each package that is registered with the cluster manager. In this directory you will find three files:

- **Configuration file (.conf)** — Contains configuration details for the package
- **Control file (.scr)** — Contains script for start, stop, and monitor functions; Progress Software Corporation recommends that you do not edit this file
- **Log file** — Created when the resource is brought online for the first time; once created, the file remains there

Restarting on the same node for HPUX

A package that fails, attempts to switch to the next node on which it is configured to run. If you want the package to restart on the node where the failure occurred, you must use the MC/ServiceGuard `cmmodpkg` utility to re-enable the package to run on that node. See your MC/ServiceGuard documentation for details regarding the syntax of this utility.

Sun Cluster uses global devices

Clusters only supports databases stored on global devices. The Sun Cluster uses global devices to provide cluster-wide, highly available access to any device in a cluster, from any node, without regard to where the device is physically attached. In general, if a node fails while providing access to a global device, the Sun Cluster software automatically discovers another path to the device and redirects the access to that path. The local disks on each server are not multi-ported, and thus are not highly available devices. Multi-ported global devices provide more than one path to a device. In the case of multi-host disks, because the disks are part of a disk device group hosted by more than one node, the multi-host disks are made highly available.

Caution: The length of the mount point of the database cannot exceed 18 characters

Windows Server 2008 R2 Cluster support with OpenEdge Windows 64-bit

Due to changes in Windows Failover Cluster implementation, there are minor differences in the OpenEdge support for Windows 2008 R2.

In Windows 2003, OpenEdge uses the default group "Cluster Group" to register the RDBMS cluster resources. In Windows 2008 R2, Microsoft changed "Cluster Group" to be a Core group that is hidden and not able to be viewed or manipulated in the Failover Cluster Manager. A new group, "OpenEdgeDatabase" is added when an OpenEdge database is enabled for clustering. This group is available through the Failover Cluster Manager for you to administer your database and other resources. When you enable a database for clustering for the first time, the "OpenEdgeDatabase" group is added to Services and Applications. The OpenEdge database and its dependent disks are resources of this group. If required, you can add more resources, for example network name, to this group using the Windows Cluster Administrator.

Using a cluster-enabled database with the AdminServer

Configuring a cluster-enabled database for use with the AdminServer is a two-step process. First the AdminServer must be started in a manner that identifies that it is running in a cluster, and then the database must be configured through OpenEdge Management or OpenEdge Explorer to indicate that it is in a clustered environment.

Start the AdminServer in a cluster

Enter the following command:

```
proadsv -start -cluster -host cluster-alias
```

Note: In an unclustered environment, PROADSV binds to the hostname `localhost`. However, in a clustered environment, `localhost` may not be available so the AdminServer must bind to the cluster alias.

Next configure a cluster-enabled database to run under the AdminServer:

1. Verify that the database resides on shared disk resources that are available to the currently active node in the cluster.
2. Start OpenEdge Management or OpenEdge Explorer and connect to the AdminServer for the cluster.
   a) From the list frame on the left, choose the database container.
   b) Select **OpenEdge** > **Database**.
   c) Select the database you want to configure to run under the AdminServer.

d) Click **Configuration**.

e) Select the **Cluster enabled** check box.

Selecting the **Cluster enabled** check box indicates you have already configured the database as cluster-enabled using the PROCLUSTER commands. This allows the database to register with the cluster manager at start time.

3. Update the database parameter file `dbname.pf` that resides in the same directory as the database `.db` file.

   The file should contain the following:

```
-cluster protected
-properties /usr/dlc/properties/conmgr.properties
-servergroup database.defaultconfiguration.defaultservergroup
-m5
-adminport adminserverport# substitute the port name or number
```

4. Start the database using OpenEdge Management or OpenEdge Explorer.

Using a cluster-enabled database with standard commands

A cluster-enabled database can be started with the standard database commands, but a special parameter is required to ensure that it is started in a cluster-protected mode. The following command starts a cluster-enabled database using PROSERVE:

```
proserve -db dbname -cluster startup
```

The `-db` and `-cluster` parameters are the only parameters allowed; any others are ignored. The effect of this command is that `proserve` directs the cluster manager to start the database using the configuration provided when the database was enabled for failover. See PROSERVE command on page 467 for a complete description of the PROSERVE command. See Cluster-enabling a database on page 243 for information on enabling a database.

Using the Windows Cluster Administrator

The Windows Cluster Administrator is a powerful and intuitive graphical user interface that allows a user with administrator privileges the ability to tailor the cluster environment. Once you open a connection to your cluster, the administrator browser displays **Groups**, **Resources**, **ClusterConfiguration**, and the member nodes as the primary selection objects. Each of the objects can be expanded to display their contents much like Windows Explorer. Resources can be moved from group to group simply by dragging and dropping them. For more information on using the Cluster Administrator, see your Microsoft documentation. The procedures that follow detail how to create and manage database resources with the Windows 2003 Cluster Administrator.

Note: The Cluster Administrator for Windows 2008 R2 has a very different user interface. See Windows Server 2008 R2 Cluster support with OpenEdge Windows 64-bit on page 249 for a description of the changes.

Use the following to create a database resource with Cluster Administrator:

1. Open a connection to your cluster and select **New** > **Resource** for the group where you are adding the database.
2. Fill in the dialog box as follows:
   a) Fill in **Name** with the name of the resource. The fully qualified name of the database is the recommended value.
   b) Fill in **Description** with anything meaningful to the user.
   c) Select OpenEdge Database for **Resource type**.
   d) **Group** should default to the one you selected in Step 1 on page 251.
   e) Check **Run this resource in a separate resource monitor**.
3. Select the preferred node default of any node.
4. Select the disk(s) and cluster name (or IP address) as resources the database depends on.
5. Fill in the **Parameters** table for the database resource as follows:
   a) Enter the fully qualified name of the database for **Database File Spec**.
   b) Enter the fully qualified path to the working directory of your OpenEdge install for **Working Directory**.
   c) Enter the name of the command to start the database for **Start Command**. For example, `C:\Progress\OpenEdge\bin\_mprosrv`. Enter only the name of the command, not the complete command line to execute.
   d) Supply the options normally specified on the command line in **Start Options.** For example, `-pf <parameter-file>`.
   e) Enter the command used to shutdown the database in **Stop Command**. For example, `C:\Progress\OpenEdge\bin\proshut`. Enter only the name of the command. Do NOT include the database file specification.
   f) Enter the options normally used to shutdown the database in **Stop Options**. For example, `-by`.
6. Select **Finish** to complete the operation. The database resource will be displayed in the Explorer window in the **Offline** state.

Once the database resource is created, it can be started and shut down through the Cluster Administrator.

Starting a database resource with the Cluster Administrator

To start the database, right click on the database resource and select **Bring Online**. The state of the database resource will change from **Offline**, to **Online Pending**, to **Online** in the Explorer window as the database completely starts up.

Shutting down a database resource with the Cluster Administrator

To Shut down the database, right click the resource name and select Take Offline. The state of the database will change from **Online**, to **Offline Pending**, to **Offline** once the database has completely shut down.

Adding a secondary broker

To add a secondary broker to a cluster-enabled database with Cluster Administrator:

1. Select **Resources** from the tree view.
2. Select your database resource.
3. Select **File** > **New** > **Resource**. The **New Resource** dialog box appears.
4. Proceed through the configuration of the new resource:
   a) For **Name**, enter the name you want for the secondary broker resource.
   b) For **Resource type**, chose Progress Database.
   c) For **Possible owners**, chose the same hosts as you selected for your database.
   d) For **Dependencies**, chose your database.
5. When you have completed the configuration, bring up the **Properties** dialog box, and fill in the **Parameters** table for the database resource as follows:
   a) Enter the fully qualified name of the database for **Database File Spec**.
   b) Enter the fully qualified path to the working directory of your OpenEdge install for **Working Directory**.
   c) Enter the name of the command to start the database for **Start Command**. For example, `C:\Progress\OpenEdge\bin\_mprosrv`. Enter only the name of the command, not the complete command line to execute.
   d) Supply the options normally specified on the command line to start a secondary broker in **Start Options.** For example, `-pf start_secondary_brkr.pf`.
   e) Leave **Stop Command** and **Stop Options** blank. Shutdown of the database is handled by the commands specified with the primary broker.

If you want to start more than one secondary broker, modify the procedure to specify a `.bat` file for the **Start Command** entered in Step 5. The `.bat` file must contain the command lines to start the secondary brokers.

Emergency disabling of a cluster-enabled database

In the event of an unrecoverable error, you can clear the cluster setting in the master block of a cluster-enabled database. Use the following command to clear the cluster setting from the master block:

```
prostrct cluster dbname clear
```

Clearing the cluster setting allows you to start and manage the database without cluster protection. Clearing the cluster setting does not clean up cluster-specific objects associated with the database; you must manually remove these objects.

Note: Use of the PROSTRCT command to clear the cluster setting is for emergencies only. Under normal circumstances, the PROCLUSTER command should be used to disable a cluster-enabled database. See Disabling a cluster-enabled database on page 244.

UNIX cluster management commands

Use the commands listed in the table below to manage your cluster resources with your UNIX cluster manager. For information on any of these commands, see your cluster manager documentation.

Table 43: UNIX cluster management commands

Generic function	AIX	HPUX	Solaris
Default script generation	n/a	`cmmakepkg`	n/a
Enumerate cluster objects	`clfindres`, `clgetgrp`, `clgetaddr`, `cllscf`	`cmquerycl`, `cmgetconf`, `cmscancl`, `cmviewcl`, `cmviewconf`	`scha_cluster_open`, `scha_resource_open`, `scha_resourcetype_open`
Validate configuration	`clverify_cluster`	`cmcheckconf`	n/a
View cluster or resource group status	`clstat clRGinfo`	`cmviewcl`	`scstat`
Register resource	`cl_crlvfs`, `cl_mkvg`, `cl_mkgroup`, `cl_mklv`, `cl_updatevg`	`cmapplyconf`	`scha_control`, `scha_resource_setstatus`
Remove resource	`cl_rmfs`, `cl_rmgroup`, `cl_rmlv`, `cl_updatevg`	`cmdeleteconf`	`scha_control`, `scha_resource_setstatus`
Start resource	n/a	`cmrunpkg`	`scha_control`, `scha_resource_setstatus`
Stop resource	n/a	`cmhaltpkg`	`scha_control`, `scha_resource_setstatus`
Move resource to node	n/a	`cmhaltpkg`, `cmrunpkg`	`scha_control`, `scha_resource_setstatus`

For AIX and HP, shared disks must be made available for use, *varied-on*, and mounted. Use the commands listed in the table below to vary-on shared resources. For information on any of these commands, see your cluster manager documentation.

Table 44: UNIX Shared disk commands

Generic function	AIX	HPUX
Vary-on volume group	`varyonvg` *`volgrp`*	`vgchange -a y` *`volgrp`*
Mount shared file system for general use	`mount/sharedfs`	`mount/sharedfs`
List volume groups	`lsvg -o`	`vgdisplay`

13

Distributed Transaction Processing

Distributed transactions involve two or more databases in a single transaction.

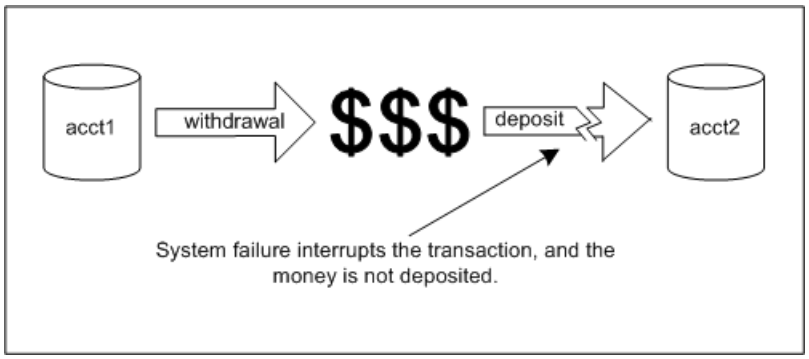
For details, see the following topics:

- Distributed transactions
- Two-phase commit with ABL clients
- Java Transaction API (JTA) support

Distributed transactions

A *distributed transaction* is a single transaction that updates two or more databases. The following scenario illustrates how inconsistencies can occur during a distributed transaction. A bank has two accounts, one on database `acct1` and another on database `acct2`. The bank runs an application that starts a transaction to withdraw a sum of money from `acct1` and deposit it into `acct2`. To keep the accounts in balance, it is critical that both operations—the withdrawal and the deposit—succeed, or that they both fail. For example, if `acct1` commits its part of the transaction and `acct2` does not, there is an inconsistency in the data, as shown in the following figure.

Figure 26: Data inconsistency



Two-phase commit with ABL clients

Two-phase commit ensures that distributed transactions occur consistently across all databases. Two-phase commit protects against inconsistency by making sure that all databases commit the transaction, or that none commit. To ensure database integrity across all involved databases, the database engine commits database updates in two distinct phases. During the first phase, the database engine checks each database involved in a transaction to verify that it is ready to commit the transaction. During the second phase, the database engine directs the databases to commit the transaction and then verifies that they committed it properly. If there is an inconsistency, the database engine displays error messages and allows you to complete or roll back the inconsistent transaction to return the data to a consistent state.

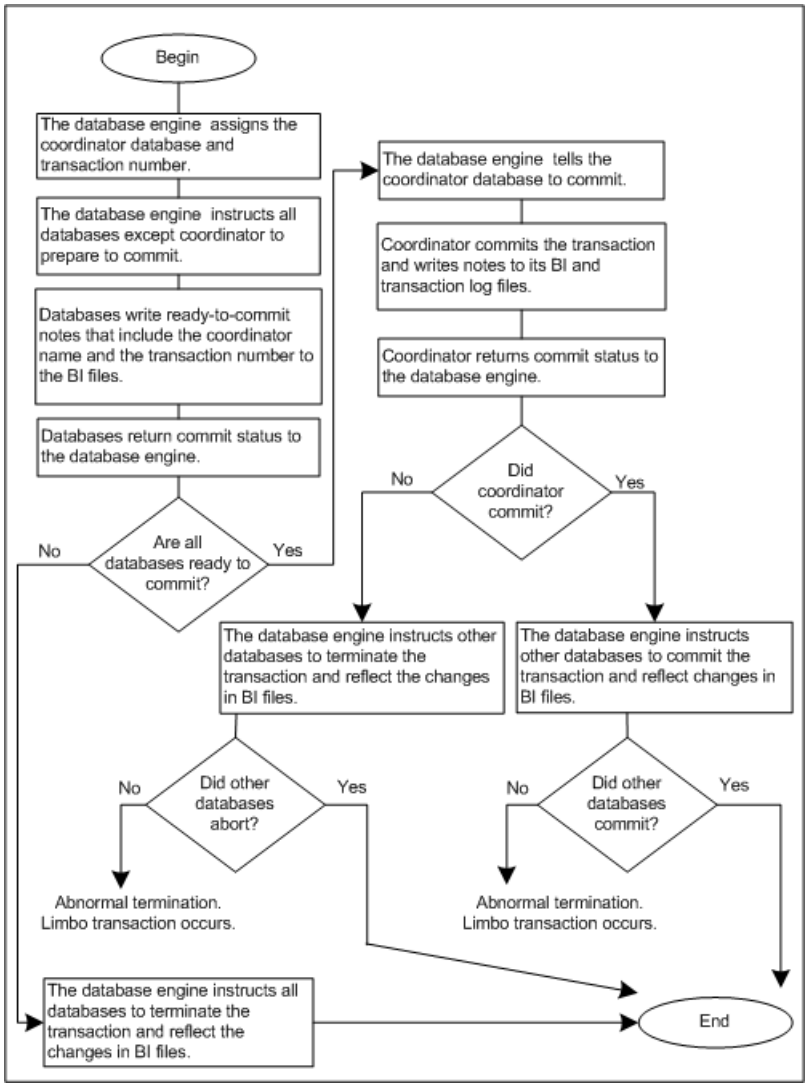
How the database engine implements two-phase commit

To implement two-phase commit, the database engine assigns a coordinator database and a transaction number for each distributed transaction. The *coordinator database* establishes the transaction's status (either committed or terminated). The *transaction number* identifies individual transactions. The transaction number for a distributed transaction is the transaction number assigned to the coordinator database. The database engine stores the name of the coordinator database and the transaction number in the before-image (BI) file of each database involved in the distributed transaction.

The coordinator database establishes the status of the transaction by committing or terminating the transaction. The action that the coordinator database takes is final and irreversible. If the coordinator database commits a transaction, the status of the transaction is committed, even if other databases do not commit the transaction. (This can happen because of a hardware or software failure.) Likewise, if the coordinator terminates a transaction, the status of the transaction is aborted. All of the other databases involved in the transaction must perform the same action as the coordinator database, or your data will be inconsistent.

The following figure shows the algorithm that the database engine uses to implement two-phase commit. Because this algorithm requires additional unbuffered I/O operations to ensure transaction integrity, there might be a performance impact when you implement two-phase commit. If two-phase commit is enabled, transactions that do not involve multiple databases do not incur additional I/O for those transactions.

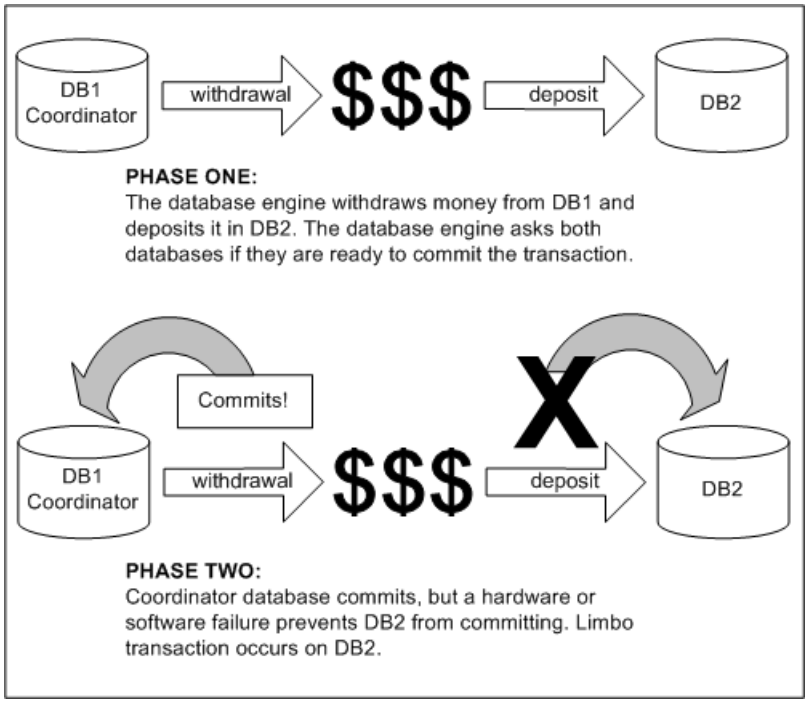
Figure 27: Two-phase commit algorithm



As the figure above shows, the coordinator database is the first database in the distributed transaction to either commit or abort the transaction. By keeping track of the coordinator's action, two-phase commit allows you to resolve any inconsistent transactions.

A *limbo transaction* (also known as an in-doubt transaction) occurs if the coordinator database commits or aborts a distributed transaction, but a hardware or software failure prevents other databases from doing likewise. This is called a limbo transaction because the processing of the transaction is temporarily suspended. A limbo transaction might occur for a variety of reasons; for example, as a result of a power outage. The following figure illustrates a limbo transaction.

Figure 28: Limbo transaction



When a limbo transaction occurs, you must resolve the transaction to re-establish data consistency.

Once the coordinator database establishes the status of a distributed transaction, it writes the status to its BI and TL (transaction log) files. The TL file tracks the status of all distributed transactions that affect the coordinator database.

Since the database engine continually overwrites the contents of the BI file, the TL file is necessary to permanently record the status of a transaction. If you must resolve limbo transactions, the transaction log file ensures that the coordinator has a reliable record of the transaction.

If you enable after-imaging for the coordinator database, the coordinator automatically uses the after-image (AI) file to log the status of each distributed transaction, instead of using the transaction log file. By using the AI file, the database engine writes to disk less often than if you use both the AI and transaction log file, thus improving performance. However, the database engine still uses the transaction log file to store information when you make an AI file available for reuse. In addition, if you disable after-imaging, the coordinator once again uses the transaction log file.

Two-phase commit and roll-forward recovery

If you use two-phase commit, you should also use after-imaging to ensure database integrity and avoid backup synchronization problems. Although two-phase commit ensures that distributed databases remain synchronized, it does not protect you from a lost database or BI file. If you lose an entire file or disk, you must use the AI file and roll-forward recovery to return to the point of the crash.

Keep the following information in mind when performing roll-forward recovery using RFUTIL:

- If you perform roll-forward recovery on a database that has after-imaging and two-phase commit enabled, RFUTIL disables after-imaging and two-phase commit.

- When you roll forward an after-image file that contains coordinator transaction end notes, RFUTIL writes a transaction log file containing the notes. Also, if two-phase commit is not enabled, RFUTIL enables two-phase commit for the coordinator database.

See Recovering a Database on page 117 for more information about roll-forward recovery and after-imaging.

Enabling two-phase commit

PROUTIL provides two-phase commit protection only if you enable two-phase commit on two or more of the databases involved in a distributed transaction. For example, if a transaction involves three databases and you enable two-phase commit for two of them, PROUTIL provides two-phase commit protection for the two databases. However, PROUTIL protects only the databases that you enable, so the transaction is not completely protected from failure. For true integrity, enable two-phase commit for all three databases.

Note: You must create and maintain a transaction log (TL) area for your database in order to use two-phase commit. For more information, see Transaction log area on page 260.

You enable two-phase commit with the PROUTIL 2PHASE BEGIN qualifier. When you enable two-phase commit, you can specify the database that should serve as the coordinator database. You can also specify an alternate name (nickname) for the coordinator database.

The syntax for enabling two-phase commit is as follows:

```
proutil db-name -C 2phase begin [ -crd | -tp nickname ]
```

The `-crd` parameter specifies that the database can serve as a coordinator database. For example, if you enable two-phase commit for three databases (`db1`, `db2`, and `db3`) and you specify the `-crd` parameter for `db3`, PROUTIL assigns `db3` as the coordinator database. However, if you specify the `-crd` parameter for more than one database, PROUTIL arbitrarily assigns a coordinator database from the databases that received the `-crd` parameter. If you do not assign any database as a coordinator, all two-phase-commit-enabled databases are potential coordinator databases. PROUTIL randomly assigns a coordinator database from one of these databases.

Specify a unique nickname for the coordinator database with the `-tp` parameter. If you do not specify a nickname, PROUTIL automatically chooses the name of the database (without the `.db` extension) as the nickname. For example, if you have a database named `/usr/dbs/appl.db`, the nickname for the database is `appl`. If PROUTIL assigns `appl.db` as the coordinator database, it writes the nickname `appl` to the BI file instead of the database's full path name. Specify nicknames of up to eight characters. Specifying a shorter nickname decreases the size of the notes that must be written.

Note: Be sure to specify a unique nickname. If you must resolve limbo transactions with two databases that have the same path name but are on different machines, PROUTIL does not distinguish between the two databases.

Modifying the database nickname and priority

If you want to change the nickname of a database, or if you want to change the priority of your databases, use PROUTIL with the 2PHASE MODIFY qualifier:

```
proutil db-name -C 2phase modify [ -crd | -tp nickname]
```

When you specify -crd, PROUTIL toggles whether or not the database can serve as a coordinator database. If you specify -crd against a database that is a candidate for coordinator database, it is no longer a candidate. If you specify -crd against a database that is not a candidate, it becomes a candidate.

When you specify -tp *nickname*, PROUTIL identifies a new nickname for the coordinator database.

Transaction log area

A separate transaction log (TL) storage area holds the transaction log data generated when two-phase commit is in use. You must create and maintain a TL area for your database in order to use two-phase commit.

The transaction log contains the transaction number of all committed distributed transactions. When there is an in-doubt transaction, this log is scanned for the transaction number to determine if the transaction committed or not. If the transaction number is found in the log, it means the transaction committed. If the transaction number is not found in the log, it means the transaction aborted.

When the last block of the TL area is written, PROUTIL automatically resets the current write position to the first block of the area. Thus, the oldest transaction recorded in the log is in the block after the current write position once the write pointer has wrapped around for the first time. The record of the transaction commit written to the log is the 32-bit transaction id of the committed transaction. Aborted transactions do not have a record written to the log.

The TL area can be composed of one or more fixed-length extents. The TL area cannot have a variable-length extent. The TL block size is 16K. Each 16K block can contain the commit record for 4,000 distributed transactions. The transaction log should contain enough space so that the record of a committed transaction is not overwritten before a database containing an in-doubt transaction can be brought back online.

To determine how large the TL area should be, use the PROMON utility to determine the average number of transactions in a 24-hour period. Multiply that number by 1.2, then round up to the nearest multiple of 16KB. The result is the number of bytes you should specify for your TL extents.

For example, if a site commits 1,000,000 transactions per day, the TL extent size needed is calculated as shown:

```
1,000,000 * 1.2 = 1,200,000
1,200,000/16384 = 73.242 (rounded up to 74)
74 * 16384 = 1212416
1212416/1024 = 1184 KB
```

This indicates that the total extent length of the TL area should be set to 1184K.

Deactivating two-phase commit

The database cannot be in use when you deactivate two-phase commit. To deactivate two-phase commit for a database, use the 2PHASE END qualifier of the PROUTIL utility:

```
proutil db-name -C 2phase end
```

When you deactivate two-phase commit, PROUTIL places a note in the database log file. However, PROUTIL does not delete the database's transaction log file.

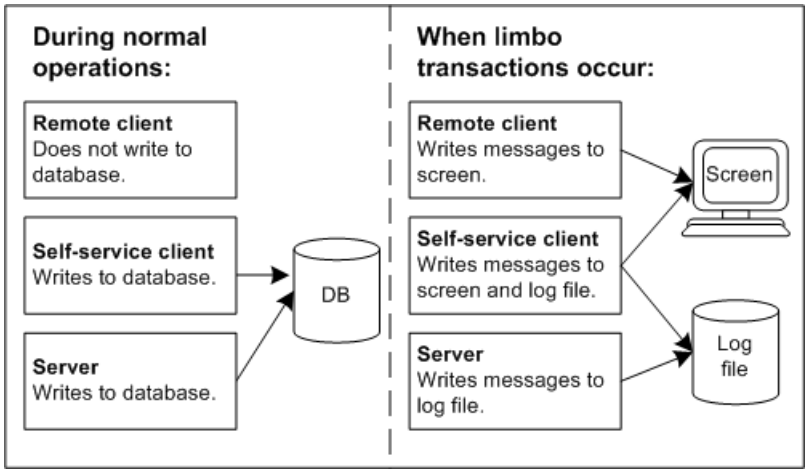
Limbo transactions with two-phase commit

A limbo transaction occurs when the synchronization between the coordinator database and other two-phase database(s) involved in a distributed transaction is interrupted. If the coordinator database commits or rolls back a transaction, and the other database do not complete their commit or roll back of the transaction, the transaction is in limbo. When a limbo transaction occurs, PROUTIL writes information about the failed transactions to the log (`.lg`) file and displays one of the following messages:

```
Part of the distributed transaction might have failed.(2022)
Cannot roll back a limbo transaction.(2030)
```

The following figure illustrates where messages are displayed.

Figure 29: How PROUTIL processes react to limbo transactions



Limbo transactions can occur without any messages being displayed on screen; for example, if a hardware or software failure occurs while a user is running a PROUTIL application or if a user powers off a client machine. If possible, users on client machines should inform the system administrator when these events occur. If such an event occurs, examine all of the databases that might be involved to determine whether any limbo transactions occurred. You can use PROMON or PROUTIL to examine a database for limbo transactions.

Caution: If an application is performing a distributed transaction when a client machine fails or shuts down, the transaction remains open. If this continues unchecked, the BI files of the databases involved in the transaction could grow to fill the disk, as with any other long-running transaction.

Resolving limbo transactions

Before you can resolve a limbo transaction, you must determine the transaction numbers, which database is the coordinator, whether the limbo transaction occurred in the coordinator database, and if the database in which the limbo transaction occurred is in use or shut down. How you resolve limbo transactions depends on whether the database is in use. If a server is running against the database, use PROMON. If no server is running, use PROUTIL.

To resolve a limbo transaction with PROMON:

1. Determine whether one or more limbo transactions occurred against a database by starting the PROMON database monitor.

Enter the following command:

```
promon db-name
```

When you enter the PROMON utility, the main menu appears:

```
  OpenEdge MONITOR Release 11
  Database: /usr/dlc/sports
  1.  User Control
  2.  Locking and Waiting Statistics
  3.  Block Access
  4.  Record Locking Table
  5.  Activity
  6.  Shared Resources
  7.  Database Status
  8.  Shut Down Database
  9.  Currently Connected Tenants
  R&D.  Advanced Options
  T.  2PC Transactions Control
  L.  Resolve 2PC Limbo Transactions
  C.  2PC Coordinator Information
  J.  Resolve JTA Transactions
  M.  Modify Defaults
  Q.  Quit
  Enter your selection:
```

2. Choose option **T (2PC Transaction Control)**.

   PROMON displays a screen similar to the following:

```
  1.  Display all entries
  2.  Match a user number
  3.  Match a range of user numbers
  5.  Change list sorting        - by user number
  6.  Match a tenant or tenants  - None
  Q.  Return to main menu
  Enter your selection:
```

3. Choose **1. (Display all entries)**.

   PROMON displays a screen similar to the following:

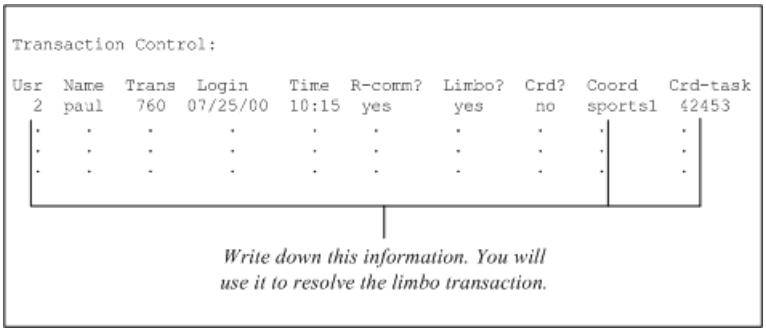
```
Transaction Control:

 Usr  Name  Trans  Login    Time  R-comm?  Limbo?  Crd?  Coord    Crd-task
   2  paul   760  07/25/00  10:15  yes      yes     no    sports1  42453
   .    .     .       .       .     .       .       .     .        .
   .    .     .       .       .     .       .       .     .        .
   .    .     .       .       .     .       .       .     .        .
```

A limbo transaction displays yes in this field.

Note: If you run PROMON against a database where no limbo transaction has occurred, PROMON does not display any field information on the **Transaction Control** screen.

Take note of any limbo transactions. For example:



A transaction is in limbo if **yes** is displayed in the **Limbo** field. For each limbo transaction, write down the following information:

- The user number, shown in the **Usr** field
- The name of the coordinator database, shown in the **Coord** field
- The transaction number of the transaction in the coordinator database, shown in the **Crd-task** field

You need this information to resolve the limbo transaction.

To resolve limbo transactions, you must consult the coordinator database of each transaction to see if the coordinator committed the transaction. If the coordinator database committed the transaction, you must also commit the transaction on the database where the limbo transaction occurred. If the coordinator did not commit the transaction, you must terminate the transaction on the database where the limbo transaction occurred.

4. For each limbo transaction, run PROMON against the coordinator database to determine whether the coordinator committed the transaction.
5. From the PROMON main menu, choose **C (2PC Coordinator Information)**.

   PROMON displays a screen similar to the following:

```
OpenEdge MONITOR Release 11

   Database: /users/sports1

     Q. Quit

Enter the transaction number you want to find out if committed:
```

Note: If the coordinator database is shut down and you cannot run PROMON against it, you must use the 2PHASE COMMIT qualifier of PROUTIL to determine whether it committed the transaction.

6. Enter the transaction number that you recorded from the **Crd-task** field in , and press **RETURN**. PROMON displays a message that tells you whether the transaction committed.

Note: To commit transactions on a database that is shut down, you must use the 2PHASE RECOVER qualifier of PROUTIL.

7. Run PROMON against the database where the limbo transaction occurred to commit or abort each limbo transaction.
8. From the PROMON main menu, choose **L (Resolve 2PC Limbo Transactions)**.

   The following menu appears:

```
                1  Abort a Limbo Transaction
                2  Commit a Limbo Transaction
                Q  Quit
          Enter choice>
```

9. Commit or abort the transaction.
   a) To commit the transaction, choose **2 (Commit a Limbo Transaction)**. PROMON prompts you to enter the user number you recorded in , then press **RETURN**.

      PROMON displays a message similar to the following:

```
User 1: commit transaction and disconnect.
```

   b) To abort the transaction, choose **1 (Abort a Limbo Transaction)**. PROMON prompts you to enter the user number of the transaction you want to abort. Enter the user number, then press **RETURN**.

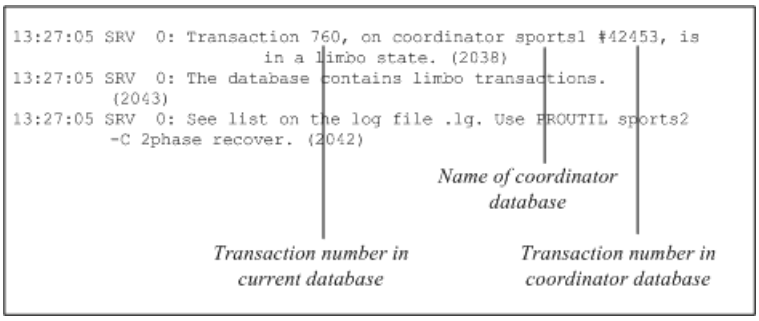
Repeat through for all the limbo transactions. After you commit or abort all of the limbo transactions, they are resolved.

Resolving limbo transactions with PROUTIL

To resolve limbo transactions using PROUTIL:

1. Try to start a database session with PROSERVE, PRO, or PROUTIL.

   f the session starts successfully, no limbo transactions have occurred on the database. If limbo transactions occurred, the session fails to start and output similar to the following is displayed, or written to the event log file for the database:



2. Capture information, for all the listed limbo transactions.

   Capture the following:

   - The transaction number on the current database (that is, the database where you tried to start the session)
   - The name of the coordinator database
   - The transaction number in the coordinator database

   Once you have this information, you must consult the coordinator database to determine whether it committed or aborted the transaction.

3. Determine if the coordinator database committed or aborted the limbo transaction.

   Enter the following command against the coordinator database to make the determination:

   ```
   proutil db-name -C 2phase commit tr-number
   ```

   In this command:

   - *db-name* specifies the coordinator database
   - *tr-number* specifies the number of the transaction to check

   Specify the number of the transaction on the coordinator database.

   If the coordinator committed the transaction, PROUTIL displays a message similar to the following:

   ```
   Transaction 42453 has committed. (2048)
   ```

   If the coordinator database committed the transaction, you must also commit the transaction on the database where the limbo transaction occurred. If the coordinator did not commit the transaction, you must abort the transaction on the database where the limbo transaction occurred.

4. Commit or abort the limbo transactions, depending on whether the coordinator committed or aborted the transaction in .

   Use the PROUTIL 2PHASE RECOVER utility to commit or abort transactions for a database. Before you enter this command, determine whether you will commit or abort each transaction; you must either commit or abort all limbo transactions to complete this command:

   ```
   proutil db-name -C 2phase recover
   ```

When you run this command against a database with limbo transactions, PROUTIL displays a message similar to the following:

```
Commit transaction 760, on coordinator sports1 #42453
    (y to commit/n to abort)? (2039)
```

5. If you respond `yes`, PROUTIL commits the transaction. If you respond `no`, PROUTIL aborts the transaction.

   PROUTIL displays this message for all of the limbo transactions that exist in the database.

After you commit or abort all of the limbo transactions, they are resolved.

Resolving limbo transaction scenarios

This section describes three scenarios in which you must resolve limbo transactions.

Scenario 1: You are on a client machine and the server fails

If you are on a client machine running a distributed transaction and something goes wrong with the server, the following message is displayed:

```
Part of the distributed transaction may have failed. (2022)
```

This message does not necessarily mean that a transaction failed. Occasionally, a transaction commits properly, but a network communication failure intercepts the server's message verifying that it committed. When you see this message, or any similar message, the database administrator must determine whether a limbo transaction occurred, then resolve the limbo transaction.

To resolve limbo transactions, you complete the transactions from the point where they were interrupted by the hardware or software failure. If the coordinator committed the transactions, you must commit the transactions. If the coordinator did not commit the transactions, you must abort the transactions.

Scenario 2: You are starting up PROUTIL and have a power failure

You have a power outage that shuts down all of the machines on your network. When the power is restored, you try to start a database session. If the session fails to start, a message similar to the following is displayed:

```
13:27:04 SRV  0: Multi-user session begin. (333)
13:27:05 SRV  0: The database contains limbo transactions. (2043)
13:27:05 SRV  0: See list on the log file .lg. Use  sports2 -C 2phase   recover. (2042)
13:27:06 ** The server terminated with exit code 20. (800)
```

This message indicates that limbo transactions must be resolved. Consult the log file for a record of the limbo transactions.

Scenario 3: You are on a client machine and it fails

Suppose a hardware or software failure occurs on a running client machine, or a user inadvertently powers off a machine while the database is running. A message indicating that a limbo transaction occurred cannot be displayed, since the client machine is down. In this situation, use the PROMON utility against the server to determine whether any limbo transactions occurred. If so, resolve them.

Two-phase commit case study

This case study illustrates the process of resolving a limbo transaction. It involves two databases, `sports1` and `sports2`, located on separate machines, `mach1` and `mach2`, respectively. Each database has two-phase commit enabled and each has a server running against it. The coordinator database is `sports1`.

Suppose that you start a client process on `mach1` against the `sports1` database and then connect to the `sports2` database using the following command:

```
CONNECT sports2 -H mach2 -S sportssv
```

After connecting, you try to run a distributed transaction. While running this procedure, the client process is halted by a system error, and the following messages appear:

```
Error reading socket, ret=-1, errno=2. (778)
Part of the distributed transaction might have failed. (2022)
Press space bar to continue.
```

The message indicates that a limbo transaction might have occurred. You must determine whether a limbo transaction did occur, then resolve it.

You start PROMON against `sports1`, choose **T (Transaction Control)**, and choose **1 (Display all entries)**. The following screen appears, indicating that there are no limbo transactions on `sports1`:

```
Transaction Control:
Usr Name  Trans  Login  Time  R-comm?  Limbo?  Crd?  Coord  Crd-task

RETURN - repeat, U - continue uninterrupted, Q - quit:
```

If PROMON failed to run against `sports1`, it indicates that the server also crashed and you must use PROUTIL to determine whether any limbo transactions occurred.

After determining that no limbo transactions occurred on `sports1`, perform the same steps against `sports2`. This time, the following screen appears, indicating that a limbo transaction has occurred:

```
Transaction Control:
Usr Name  Trans  Login  Time  R-comm?  Limbo?  Crd?  Coord  Crd-task
 15 paul  755  04/01/02 14:19  yes      yes     no  sports1  61061
                      .
                      .
                      .
RETURN - repeat, U - continue uninterrupted, Q - quit
```

Write down the coordinator's transaction number (indicated in the **Crd-task** field). The **Coord** field indicates that `sports1` is the coordinator database for this transaction. Therefore, you must again run PROMON against `sports1`. This time, choose **C (Coordinator Information)**. The following screen appears:

```
PROGRESS MONITOR Release 11

   Database: /users/sports1

   Q. QUIT

Enter the transaction number you want to find out if committed: 61061
```

Enter the transaction number, `61061`. The following screen appears, indicating that the transaction committed:

```
Scan the logs...

** Transaction 61061 has committed.

Q. QUIT

Enter the transaction number you want to find out if committed:
```

Since the transaction committed on the coordinator `sports1`, you run PROMON against `sports2` and choose **1 (Resolve Limbo Transactions)**. The following screen appears:

```
Transaction Control:
Usr   PID  Time of Login            User ID  TTY         Coord  Crd-task
 15  3308  Fri Apr 5 14:19:45 2002  paul   mach1 ttyp1  sports1   61061

     1 Abort a Limbo Transaction
     2 Commit a limbo Transaction
     Q Quit

Enter choice>
```

Choose **2 (Commit a Limbo Transaction)**, and the following prompt appears:

```
Enter the user number whose transaction you want to commit:
```

Type `15` (the user number indicated on the previous screen). The PROMON utility commits the transaction on `sports2` and displays the following message:

```
User 15: commit transaction and disconnect.
```

Since there are no more limbo transactions, the situation is resolved and no further action is required.

Java Transaction API (JTA) support

Support for the Java Transaction API (JTA) in the OpenEdge SQL engine enables the OpenEdge RDBMS to participate in distributed SQL transactions. The JTA defines transaction management between a transaction manager and a resource manager within the standard J2EE framework architecture. In this scenario, OpenEdge is the resource manager and a SQL application server or client is the transaction manager. The database is **not** the ultimate authority on the distributed transaction. For details on J2EE and JTA, refer to your Java documentation.

When an OpenEdge database is configured as a resource manager for distributed JTA transactions, the transaction manager is responsible for establishing and maintaining the state of the transaction. The database receives an identifier for the global transaction context. It is possible that multiple threads of execution will process the transaction, impacting transaction processing in several ways:

- It is possible for records to be locked by JTA transaction with no user associated with the lock.
- It is possible for record locks to exist at database startup.
- Locks are owned by the transaction, not the user.

JTA resource impact

Enabling the database as a JTA resource manager increases the consumption of the following database resources:

- **After-image and before-image files** — Additional notes support JTA transactions and table lock acquisition. AI and BI files must be increased at least 30 percent when JTA is enabled.
- **Lock Table (`-L`)** — JTA transactions may hold onto locks for a longer period of time than local transactions.
- **Transaction Table** — The number of rows in the transaction table is increased by the maximum number of JTA transactions. The maximum number is controlled by the `-maxxids` startup parameter.
- **Xid Table** — An additional table for storing JTA transaction information. The size of the table is determined by the maximum number of JTA transactions allowed, and is controlled by the `-maxxids` startup parameter.

JTA processing impact

Enabling the database as a JTA resource manager alters database processing in the following ways:

- Crash recovery — Crash recovery processing executes every time a database is started, whether in single-user or multi-user mode. A JTA enabled database must perform crash recovery in multi-user mode. Attempting to perform crash recovery in single user mode results in an error if any JTA transactions exist in at startup in a prepared state.
- Roll forward recovery — After-image files written by a JTA-enabled database, must be applied to a JTA-enabled database. In addition, the `endtime` and `endtrans` options for ROLL FORWARD are not allowed.
- Before-image processing — Long running JTA transactions prevent reuse of before-image clusters.

Enabling JTA support

Prior to establishing your database as a JTA resource manager, you must enable support of JTA transactions. The following command enables your database:

```
proutil db-name-C enablejta
```

For complete syntax details, see PROUTIL ENABLEJTA qualifier on page 701.

Disabling JTA support

To remove your database from participation in distributed JTA transactions, you can disable JTA support. The following command disables your database:

```
proutil db-name-C disablejta
```

For complete syntax details, see PROUTIL DISABLEJTA qualifier on page 685.

JTA transactions and OpenEdge Replication

When a JTA-enabled OpenEdge database is enabled for site replication with OpenEdge Replication, JTA transactions are replicated from the source database to the target database or databases. JTA transactions are started against the replication source database and replicated to the replication target database by the replication agent. Manual intervention to resolve JTA transactions should only be performed on a replication source database when there is an unrecoverable catastrophic failure to JTA transaction manager. On a target database, JTA transactions can only be resolved when Replication Agent is in "pre-transition" state.

For more information on OpenEdge Replication, see *OpenEdge Replication: User Guide*. For more information on manually resolving JTA transactions, see Resolving JTA transactions on page 271.

Monitoring JTA transactions

The PROMON utility displays transaction and lock information for JTA transactions in both the **Record Locking Table** display and the **Active Transactions** display. The **Trans State** displays JTA transaction states as well as standard transaction states. The **Trans ID** field displays the internal transaction id.

Possible values for the **Trans State** column for a JTA transaction are:

- **Active JTA** — The transaction is currently executing
- **Idle JTA** — The transaction is not currently executing
- **Prepared JTA** — The transaction is prepared
- **RollbackOnly JTA** — The transaction has encountered an error
- **Committed JTA** — The transaction is in the commit process

For more information on PROMON, see PROMON Utility on page 543.

Resolving JTA transactions

Since OpenEdge relinquishes transaction control for JTA transactions, it is dangerous to intervene and manually resolve outstanding JTA transactions. Intervention could compromise the referential integrity of the database as a result. Resolve JTA transactions manually only when there has been an unrecoverable catastrophic failure to the JTA transaction manager. When necessary, use PROMON to identify and resolve JTA transactions.

To resolve a JTA transaction with PROMON:

1. Determine if one or more unresolved JTA transactions exist against a database by starting the PROMON database monitor. Enter the following command:

```
promon db-name
```

   When you enter the PROMON utility, the main menu appears:

```
  OpenEdge MONITOR Release 11
  Database: /usr/dlc/sports
  1.  User Control
  2.  Locking and Waiting Statistics
  3.  Block Access
  4.  Record Locking Table
  5.  Activity
  6.  Shared Resources
  7.  Database Status
  8.  Shut Down Database
  9.  Currently Connected Tenants

  R&D.  Advanced Options
  T.  2PC Transactions Control
  L.  Resolve 2PC Limbo Transactions
  C.  2PC Coordinator Information
  J.  Resolve JTA Transactions
  M.  Modify Defaults
  Q.  Quit
  Enter your selection:
```

2. Choose option **J (Resolve JTA Transactions)**.

   PROMON displays a screen similar to the following:

```
WARNING:  Committing or rolling back a JTA transaction can compromise
WARNING:  referential integrity.  Proceed with EXTREME caution.
     1.  Display all JTA Transactions
     2.  Rollback a JTA Transaction
     3.  Commit a JTA Transaction
     Q.  Return to main menu
     Enter your selection:
```

3. Choose **1. (Display all JTA Transactions)**.

   PROMON displays a screen similar to the following:

```
OpenEdge Release 11 Monitor
Tran Id  Usr  JTA State           XID
1492      5   JTA
Prepared         4a982a20-49b7-11da-8cd6-0800200c9a66
1494     16   JTA Active          0
```

   Take note of the **Tran Id** value for any outstanding transactions. You need this information to resolve the transaction.

4. For each outstanding JTA transaction, determine if you are going to commit or rollback the transaction.
5. If you are going to commit the transaction, select **2** from the **Resolve JTA Transactions** menu. If you are going to rollback the transaction, select **3**. You are prompted to enter the transaction id value you noted in . You are prompted to confirm your decision. The transaction commit or rollback is logged in the database log file.

   Caution:

   Manually committing or rolling back a JTA transaction can compromise the referential integrity of the database.

Repeat Step 4 and Step 5 for all the outstanding JTA transactions. After you commit or abort all of the transactions, they are resolved.

III

Maintaining and Monitoring Your Database

For details, see the following topics:

- Managing Performance
- Maintaining Database Structure
- Multi-tenant database
- Table-partitioned database
- Change Data Capture
- Diagnostic Data Collection
- Dumping and Loading
- Logged Data

14

Managing Performance

The potential for improving the performance of your OpenEdge RDBMS depends on your system. Some options might not be available on your hardware or operating system platform.

For details, see the following topics:

- Introduction to performance management
- Tools for monitoring performance
- Server performance factors
- Memory usage
- Alternate Buffer Pool
- Operating system resources
- Network Communication
- Database fragmentation
- Index use
- Database Client Notification
- Changing Database Client Notification poll time
- Virtual system tables

Introduction to performance management

The OpenEdge RDBMS relies on the following system resources to perform its work:

- **CPU** — Manipulates data and executes programs
- **Disks and controllers** — Read and write data from database
- **Memory** — Stores data so it can be accessed quickly while performing operations
- **Operating system mechanisms** — Allocate and use system resources
- **Network** — Exchanges data between client and server systems

Performance is diminished if the database cannot use these resources efficiently. Performance bottlenecks occur when a resource performs inadequately (or is overloaded) and prevents other resources from accomplishing work. The key to improving performance is determining which resource is creating a bottleneck. Once you understand your resource limitations, you can take steps to eliminate bottlenecks.

Performance management is a continual process of measuring and adjusting resource use. Because system resources depend on each other in complex relationships, you might fix one problem only to create another. You should measure resource use regularly and adjust as required.

To effectively manage performance, you must have solid knowledge of your system, users, and applications. A system that is properly tuned has sufficient capacity to perform its workload. Applications running on your system should not compete with the database for system resources. Because system and application performance can vary greatly depending on the configuration, use the information in this chapter as a guideline and make adjustments as required for your configuration.

Tools for monitoring performance

This section describes several tools that you can use to monitor the performance of an OpenEdge database. It includes information about:

- PROMON utility on page 276
- Virtual system tables on page 276

PROMON utility

The OpenEdge Monitor (PROMON) utility helps you monitor database activity and performance. PROMON Utility on page 543 documents the main PROMON options. In addition, PROMON provides advanced options (called R&D options) for in-depth monitoring of database activity and performance.

Virtual system tables

Virtual system tables (VSTs) provide ABL and SQL applications access to the same database information that you can collect with the PROMON utility. The virtual system tables, or schema tables, have no physical records until the database manager generates them at run time. This enables an ABL or SQL application to retrieve information as run-time data.

Server performance factors

The following factors can affect server performance:

- CPU usage
- Disk I/O relating to database, before-image, and after-image I/O
- Record locking
- Memory usage
- Database and index fragmentation

CPU usage

To use your system to its full potential, the CPU should be busy most of the time. An idle CPU or unproductive CPU processing can indicate a bottleneck. Use operating system utilities to monitor CPU usage.

If performance is inadequate and your CPU is idle, the CPU might be waiting for another resource. Identify the bottleneck and eliminate it so that the CPU can process work efficiently. Use PROMON to monitor database activity.

Disk I/O is a common bottleneck. For more information, see Disk I/O on page 277.

Symmetric multi-processing (SMP) systems use a spin lock mechanism to give processes exclusive access to data structures in shared memory. Spin locks ensure that only one process can use the structure at a time, but that all processes can get access to these structures quickly when they have to. However, if tuned incorrectly, SMP CPUs might spend time processing the spin locks unnecessarily instead of performing useful work.

The spin lock algorithm works as follows: When a process requires a shared-memory resource, it attempts to acquire the resource's latch. When a process acquires the latch, it has exclusive access to the resource. All other attempts to acquire the latch fail until the holding process gives up the latch. When another process requires access to the resource, it attempts to acquire the latch. If it cannot acquire the resource's latch because another process is holding it, the second process continues the attempt. This iterative process is called *spinning*. If a process fails to acquire a latch after a specified number of spins, the process pauses, or takes a *nap*, before trying again. If a process repeatedly fails to acquire a latch, the length of its nap is gradually increased. You can set the Spin Lock Retries (`-spin`) parameter to specify how many times to test a lock before napping.

To use a system of semaphores and queues to control locking, set `-spin` to zero (0).

Use the PROMON R&D **Adjust Latch Options** under **Administrative Functions** to change the spin mechanism after start up.

Disk I/O

Because reading and writing data to disk is a relatively slow operation, disk I/O is a common database performance bottleneck. The database engine performs three primary types of I/O operations:

- Database I/O
- Before-image I/O
- After-image I/O (relevant only if after-imaging is enabled)

If performance monitoring indicates that I/O resources are overloaded, try the techniques in the following sections to better balance disk I/O.

The best way to reduce disk I/O bottlenecks is to spread I/O across several physical disks, allowing multiple disk accesses to occur concurrently. You can extend files across many disk volumes or file systems.

Database I/O

Database I/O occurs when the database engine reads and writes blocks containing records to and from disk into memory. To minimize database disk I/O, the database engine tries to keep a block in memory after it reads the block the first time. The next time the engine needs that block, it can access it from memory rather than reading it from disk.

To eliminate database I/O bottlenecks, you can:

- Increase the number of database buffers
- Change the number and structure of database storage areas
- Use private read-only buffers
- Use asynchronous page writers (APWs)

Storage areas

Storage areas are the largest physical unit of a database. Storage areas consist of one or more extents that are either operating system files, or some other operating system level device that is addressed randomly. A storage area is a distinct address space, and any physical address stored inside the area is generally stored relative to the beginning of the storage area.

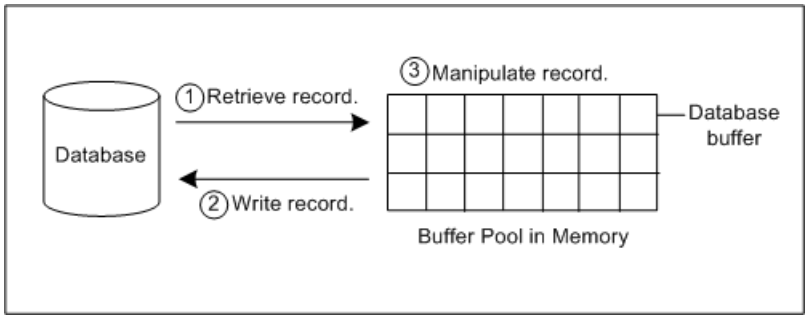
Storage areas give you physical control over the location of specific database objects. You can place each database object in its own storage area or place many database objects in a single storage area. Storage areas can contain database objects of one type or of many types. For example, to achieve load balancing, you can place a particularly active table in a separate storage area, then place the most active index for that table in its own storage area. Then, in a third storage area, place all the remaining tables and indexes. You cannot split a table or index across storage areas.

However, you can improve performance by moving tables and indexes to an application data storage area on a faster disk, while the database remains online. For a description of how to move tables and indexes while the database remains online, see Maintaining Database Structure on page 325.

Database buffers

A *database buffer* is a temporary storage area in memory used to hold a copy of a database block. When the database engine reads a database record, it stores the block that contains that record in a database buffer. Database buffers are grouped in an area of memory called the *buffer pool*. The following figure illustrates database disk I/O.

Figure 30: Database I/O

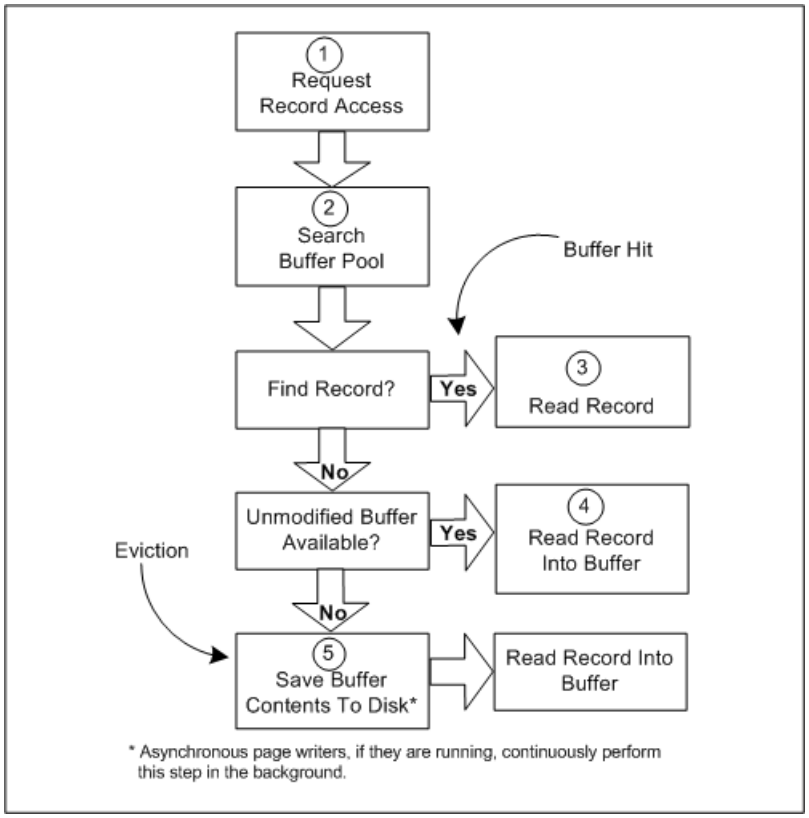


Database I/O works as follows:

1. When a process needs to read a database record, it requests access to the record.
2. The database engine searches the buffer pool for the requested record.
3. If the block that holds the record is already stored in a buffer, the engine reads the record from the buffer. This is called a *buffer hit*. When tuned correctly, the engine should achieve a buffer hit most of the time.
4. If the record is not found in any buffer, the engine must read the record from disk into a buffer. If an empty buffer is available, the engine reads the record into that buffer.
5. If no empty buffer is available, the engine must replace another buffer to make room for it.
6. If the block that will be replace has been modified, the engine must write the block to disk to save the changes. This is known as an *eviction*. While the eviction takes place, the process that requested the record in must wait. For this reason, performance is improved if empty buffers are always available. See How the database engine writes modified buffers on page 279 for detailed steps.

The following figure illustrates how the engine reads a database record into a buffer.

Figure 31: Reads and writes to database buffers

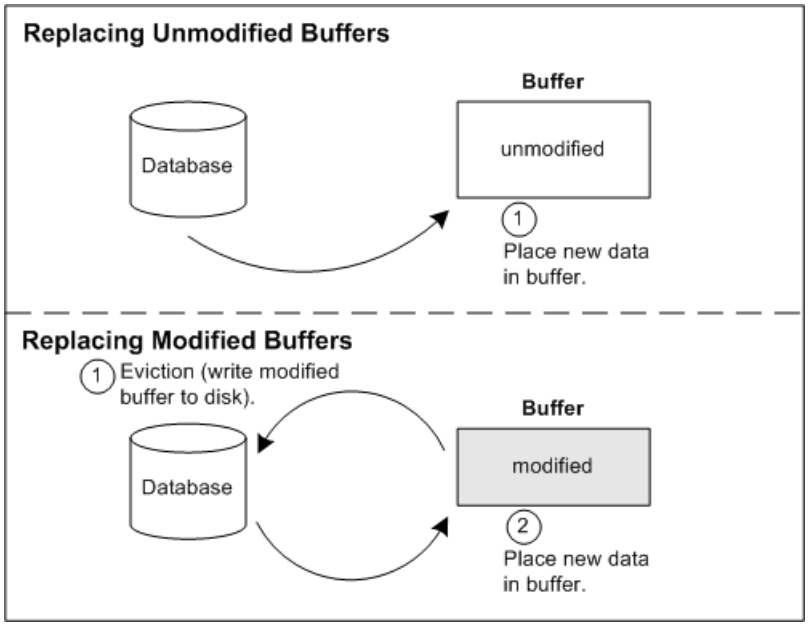


How the database engine writes modified buffers

When a process requires access to a database block that is not in the buffer pool, the database engine must replace another buffer to make room for it. The server searches for a buffer to replace.

The ideal replacement candidate is a buffer that is unlocked and unmodified. Replacing an unmodified buffer requires only one step: writing the new contents into the buffer. If a buffer contains modified data, it must first be evicted before it can be replaced. Evicting the buffer requires two steps: writing the buffer's contents to disk, then writing new contents into the buffer. It is therefore slower and requires more overhead as shown in the following figure.

Figure 32: Evicting buffers



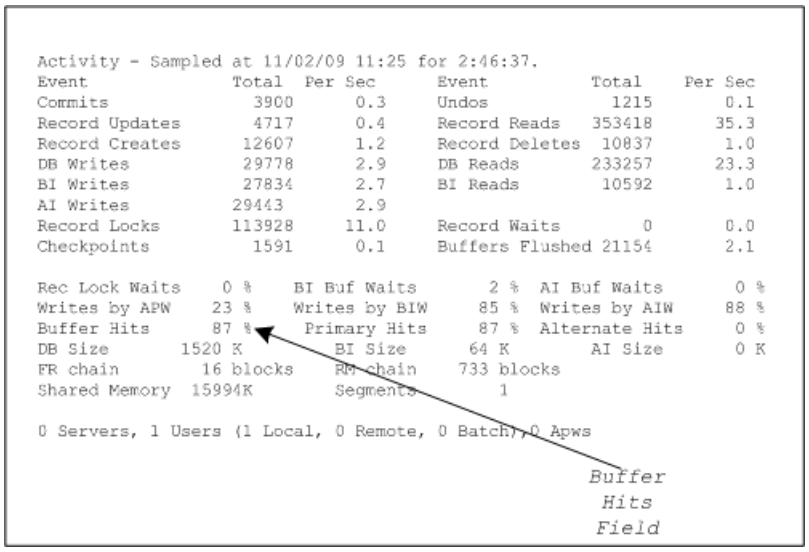
When searching for a replacement candidate, the server searches a maximum of ten buffers. If the server fails to find an unlocked, unmodified buffer, the server evicts the first unlocked, modified buffer that it finds.

Monitoring database buffer activity

A buffer hit occurs when the database engine locates a record in the buffer pool and does not have to read the record from disk. See Database buffers on page 278 for an explanation of buffer hits and how they improve performance by reducing overhead. When tuned correctly, the engine should achieve a buffer hit most of the time.

To determine the efficiency of database buffer activity, check the Buffer Hits field of the PROMON Activity option. For best performance, increase the Blocks in Database Buffers (-B) parameter until the buffer hits percentage exceeds 95 percent, or until your system starts paging. The following figure shows the Buffer Hits field in a sample Activity display.

Figure 33: Monitoring buffer activity



Tuning database buffers

If database buffer monitoring indicates that buffer hits are below 85 percent, you can increase the number of database buffers so more records are kept in memory.

To increase the number of buffer hits, increase the -B startup parameter to allocate more buffers. Increase the -B parameter until the buffer hits percentage exceeds 95 percent, or until your system starts paging.

The database engine uses a hash table to reduce the time it takes to locate a database buffer. The Hash Table Entries (-hash) startup parameter controls the number of hash table entries in the buffer pool. The database engine sets this parameter to approximately 25 percent of the number of database buffers (specified by the -B parameter). In most cases, the default value is adequate. However, increasing this parameter might slightly reduce the time required to find a block in the buffer pool.

Using private read-only buffers

The buffer pool is a mechanism that conserves I/O when multiple users are accessing information from the same disk blocks. The buffer pool has a predefined size. Once the buffer pool becomes full, buffers are replaced on a least recently used (LRU) basis. Since sequential readers of the database access so many different buffers, they sometimes monopolize the buffer pool. That is, sequential readers of the database cause many shared buffers to be replaced by the buffers most recently used by the sequential reader.

Consequently, you can request some number of buffers in the buffer pool to be private read-only buffers. Private read-only buffers do not participate in the LRU replacement algorithm of the general shared buffer pool.

Applications that read many records in a short time, such as applications that generate reports or lists, should use private read-only buffers. Private read-only buffers prevent applications from quickly using all the public buffers and depriving buffers from other users. When an application is using private read-only buffers, updates are performed correctly using the public buffers. Therefore, an application performing many read operations but only a modest amount of updates might also benefit from using private read only buffers.

When a sequential reader is using private read-only buffers and needs a buffer to perform a read operation, and the buffer is already in the private read-only buffer pool, the database engine marks the buffer as most recently used (MRU) and uses it. If the buffer is not already in the private read-only buffer pool, the sequential reader takes a buffer from the LRU chain and puts it in the private read-only buffer pool. If the sequential reader has exhausted its quota of private read-only buffers, a private read-only buffer is replaced. The sequential reader maintains a list or chain of all its private buffers and uses a private LRU replacement mechanism identical to the public-shared buffer pool LRU replacement algorithm.

All users, regular and sequential, have access to all buffers in the buffer pool (public or private). If a regular user needs a block found in a private buffer pool, the buffer is removed from the sequential readers list of private buffers and is put back into the LRU chain as the most recently used buffer. In addition, if a sequential read user needs to update a private read-only buffer, it is removed from the sequential reader's private buffer pool and put into the general shared buffer pool as most recently used.

Sequential reads use an index and require that index blocks be available in memory because they are used repeatedly. Therefore, you want to request enough private read-only buffers to hold all of the index blocks needed to retrieve a record. To determine how many private read-only buffers to set, count the number of tables that you read and determine the indexes you use. Then, determine the number of levels in the B-tree (balance tree) of each index and add 1 (for the record blocks). For example, request at least five private read-only buffers if you have a report that reads the `Customer` table using the `Cust-Name` index, and the `Cust-Name` index has four B-tree levels.

If you do not know the number of levels in your index, you can generally request six private read-only buffers and get a good result. If you perform a join and are reading from two tables simultaneously, request 12. If the system is unable to allocate the requested number of private read-only buffers, a message is written to the database log.

You can request a number of private read-only buffers using the Private Buffers (`-Bp`) startup parameter. When you use the `-Bp` startup parameter the request remains active for the entire session unless it is changed or disabled by an application. Each user of private read-only buffers reduces the number of public buffers (`-B`).

Note: The total number of private read-only buffers for all simultaneous users is limited to 25 percent of the total blocks in database buffers. This value is set by the `-B` startup parameter. See Database Startup Parameters on page 473 for information on setting `-B`.

You can also request a number of private read-only buffers from within an ABL or SQL application by setting a value in the `_MyConn-NumSeqBuffers` field of the `_MyConnection` virtual system table (VST). Since `_MyConnection` is an updatable virtual system table, private read-only buffers can be dynamically requested and released in the application. For a description of the `_MyConnection` VST, see Virtual System Tables on page 863.

The following ABL code example demonstrates how to turn private read-only buffers on and off:

```
/*Get 6 private read-only buffers for my application*/
FIND _MyConnection.
_MyConnection._MyConn-NumSeqBuffers = 6.

/**** Report using private read only buffers ***/
/* Turn off private read only buffers of my application */
FIND _MyConnection.
_MyConnection._MyConn-NumSeqBuffers = 0.
```

The following example demonstrates how to turn private read-only buffers on and off using an SQL statement:

```
UPDATE pub."_MyConnection" SET "_MyConn-NumSeqBuffers" = 6.

UPDATE pub."_MyConnection" SET "_MyConn-NumSeqBuffers" = 0.
```

Using APWs to improve performance

APWs are optional and require an Enterprise database license. APWs are highly recommended because they improve performance in the following ways:

- They ensure that a supply of empty buffers is available so the database engine does not have to wait for database buffers to be written to disk.
- They reduce the number of buffers that the engine must examine before writing a modified database buffer to disk. To keep the most active buffers in memory, the engine writes the least recently used buffers to disk; the engine must search buffers to determine which one is least recently used.
- They reduce overhead associated with checkpointing because fewer modified buffers have to be written to disk when a checkpoint occurs.

You must manually start APWs. You can start and stop APWs at any time without shutting down the database. See Starting Up and Shutting Down on page 73 for instructions on starting and stopping an APW.

A database can have zero, one, or more APWs running simultaneously. The optimal number is highly dependent on your application and environment. Start two APWs and monitor their performance with PROMON. If there are buffers being flushed at checkpoints, add an additional APW and recheck. Applications that perform fewer changes to a database require fewer APWs.

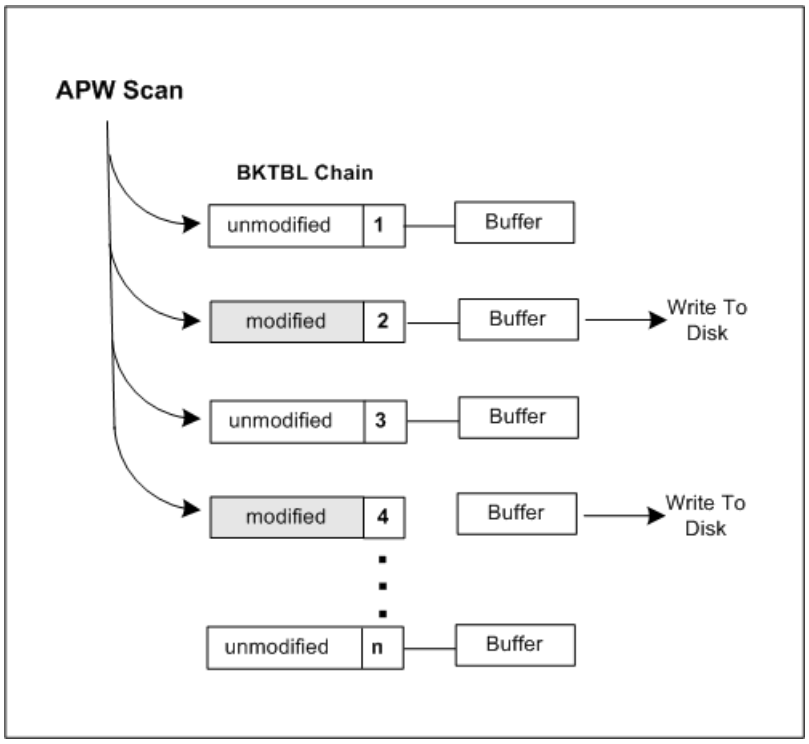
Note: If you do not perform any updates, no page writers are required.

APWs are self-tuning. Once you determine how many APWs to run, you do not have to adjust any startup parameters specific to APWs. However, you might want to increase the BI cluster size to allow them to perform at an optimal rate. PROUTIL TRUNCATE BI lets you create a BI cluster of a specific size. For more information, see PROUTIL TRUNCATE BI qualifier on page 776.

APWs continually write modified buffers to disk, making it more likely the server will find an unmodified buffer without having to wait. To find modified buffers, an APW scans the Block Table (BKTBL) chain. The BKTBL chain is a linked list of BKTBL structures, each associated with a database buffer. Each BKTBL structure contains a flag indicating whether the associated buffer is modified. When an APW finds a modified buffer, it immediately writes the buffer to disk.

The following figure illustrates how an APW scans the BLKTBL chain.

Figure 34: Block Table (BLKTBL) chain



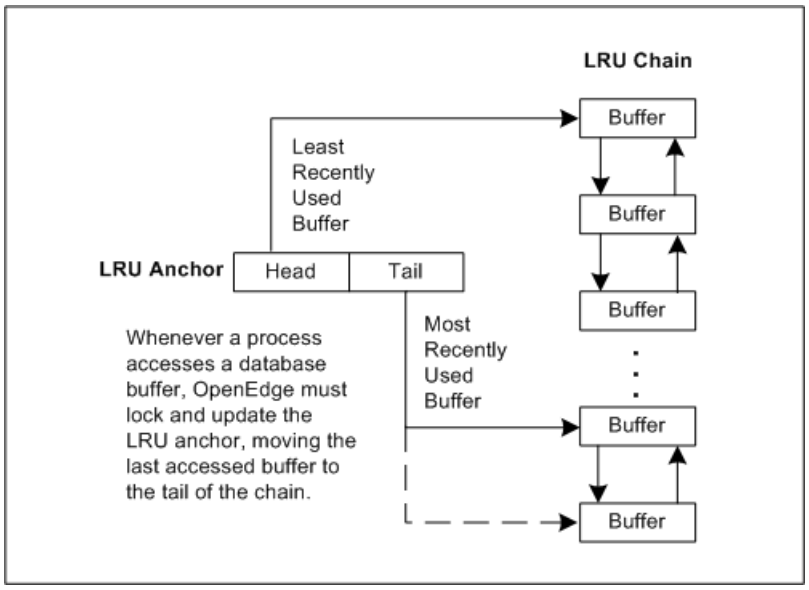
The APW scans in cycles. After completing a cycle, the APW goes to sleep. When the APW begins its next scanning cycle, it picks up where it left off. For example, if the APW scanned buffers 1 to 10 during its first cycle, it would start at buffer 11 to begin its next cycle.

When the database engine writes modified buffers to disk, it replaces the buffers in a least-to-most-recently-used order. This is beneficial because you are less likely to need older data.

To find least recently used buffers, an APW scans the least recently used (LRU) chain. The *least recently used chain* is a doubly linked list in shared memory that the engine uses to access database buffers. The LRU chain is anchored by a data structure that points to the head and tail of the chain. Whenever a process accesses a database buffer, the server must lock and update the LRU anchor, moving the accessed buffer to the tail of the chain.

The following figure illustrates the LRU chain.

Figure 35: APWs and the least recently used chain



Since all processes must lock the LRU anchor whenever they have to access a buffer, long buffer replacement searches create contention for all processes accessing the database buffer pool. This can have a debilitating effect on performance, especially on heavily loaded systems. APWs reduce contention for the LRU anchor by periodically clearing out modified buffers. When buffer replacement is required, the database engine can find an unmodified buffer quickly.

A third way that APWs improve performance is by minimizing the overhead associated with before-image checkpointing.

The before-image file is divided into clusters. A checkpoint occurs when a BI cluster becomes full. When a cluster becomes full, the database engine reuses the cluster if the information stored in it is no longer required. By reusing clusters, the engine minimizes the amount of disk space required for the BI file.

Checkpoints ensure that clusters can be reused and that the database can be recovered in a reasonable amount of time. During a checkpoint, the engine writes all modified database buffers associated with the current cluster to disk. This is a substantial overhead, especially if you have large BI clusters and a large buffer pool. APWs minimize this overhead by periodically writing modified buffers to disk. When a checkpoint occurs, fewer buffers must be written.

Monitoring APWs

The PROMON R&D option **Page Writers Activity** display shows statistics about APWs running on your system. The following figure shows a sample display.

```
01/25/00      Activity: Page Writers
16:29         from 01/25/00 13:56 to 01/26/00 11:23 (21 hrs 27 min)

                              Total   Per Min   Per Sec     Per Tx
Total DB writes               3       0         0.00        0.00
APW DB writes                 0       0         0.00        0.00
    scan writes               0       0         0.00        0.00
    APW queue writes          0       0         0.00        0.00
    ckp queue writes          0       0         0.00        0.00
    scan cycles               0       0         0.00        0.00
    buffers scanned           0       0         0.00        0.00
    bfs checkpointed          173     0         0.11        0.14
Checkpoints                   82110   0         5.22        6.79
Marked at checkpoint          0       0         0.00        0.00
Flushed at checkpoint         0       0         0.00        0.00

Number of APWs:                          1
```

Note: Nonzero numbers in the **Flushed at Checkpoint** row indicates that the APW was unable to write buffers fast enough to prevent a memory flush. Increase the number of APWs and/or increase the cluster size to eliminate the flush.

Monitoring user I/O

Table and index activity can be monitored on a per user basis. This granularity can provide insight into individual user activity, and provide data for evaluating query efficiency. The `_UserTableStat` and `_UserIndexStat` VSTs collect the data. For details on the table fields, see Virtual System Tables on page 863. You can query the VSTs directly or use PROMON to monitor the activity. For table activity, choose **PROMON** > **R&D** > **Other Displays** > **I/O Operations by User by Table**; for index activity, choose **PROMON** > **R&D** > **Other Displays** > **I/O Operations by User by Index**.

There is a cost in memory consumption to monitor I/O by user. The estimated cost is 32 bytes per table per user, and 40 bytes per index per user. Consider a 500 user system with 100 tables and 200 indexes. The total memory cost is as follows:

```
500 users * 100 tables * 32 bytes/table =
                              1,600,000 bytes of memory for tables

500 users * 200 indexes * 40/bytes/index =
                              4,000,000 bytes of memory for indexes

1,600,000 + 4,000,000 = 5,600,000 bytes, or approx 5.3 MB
```

The base and range of tables and indexes monitored by the `_UserTableStat` and `_UserIndexStat` VSTs is established at server startup. By default the first 100 objects are monitored. The table below describes the startup parameters. You can alter the base during runtime, but not the range.

Table 45: Startup parameters for I/O by object

Parameter	Indication
`-basetable n`	Start monitoring tables at table *n*. The default is 1. Change the `basetable` value during runtime by updating the value of the `_TableBase` field in the `_StatBase` VST.
`-tablerange n`	Monitor *n* tables. If not specified, the range is 100.
`-baseindex n`	Start monitoring indexes at index *n*. The default is 1. Change the `baseindex` value during runtime by updating the value of the `_IndexBase` field in the `_StatBase` VST.
`-indexrange n`	Monitor *n* indexes.[4] If not specified, the range is 100.

To consume less memory, set the `-indexrange` and `-tablerange` parameters to smaller values.

LRU Concurrency

For databases with an Enterprise license, two buffer pool tuning parameters provide the ability to ease contention on the buffer pool LRU latch by altering LRU replacement policy in both the primary and alternate buffer pools, with the goal of improving throughput performance. (See Alternate Buffer Pool on page 298 for information on the Alternate Buffer Pool.)

The LRU skip parameters allow you to specify the number of times a buffer in the buffer pool is accessed before it is placed on the Most Recently Used (MRU) end of the LRU chain. Accessing a buffer a specified number of times eliminates the need to acquire the LRU latch each time a buffer is accessed.

Specifying the LRU skip parameters is helpful when high LRU latch wait timeouts are occurring (observable in PROMON or by VST). When the LRU skip value is tuned to your environment, contention on the LRU latch decreases, improving concurrency and performance, and there is no increase in the number of page misses in the buffer pool. If increasing the LRU skip value causes increased buffer pool misses, then the value should be tuned down or turned off altogether. Setting the LRU skip value to zero disables the feature, and is the default.

LRU skip values can be specified at startup through the startup parameters, `-lruskips` and `-lru2skips`, or they can be set and adjusted with PROMON or VST. They are not implemented for private buffer pool data access.

If you modify the value after startup, the change will not take effect on buffers currently in the buffer pool until the current value expires and the buffer is moved to the MRU end of the chain.

For mor information, see:

- Database Startup Parameters on page 473 for details on specifying the LRU skip parameters at database startup.
- PROMON Utility on page 543 for details on monitoring and modifying the LRU skip parameters.

[4] You cannot alter the range after startup.

Before-image I/O

Before-imaging is always enabled to let the database engine recover transactions if the system fails. This mechanism is extremely important for database reliability, but it creates a significant amount of I/O that can affect performance. In addition, before-image I/O is usually the first and most likely cause of I/O bottlenecks. The engine must always record a change in the BI file before it can record a change in the database and after-image files. If BI activity creates an I/O bottleneck, all other database activities are affected.

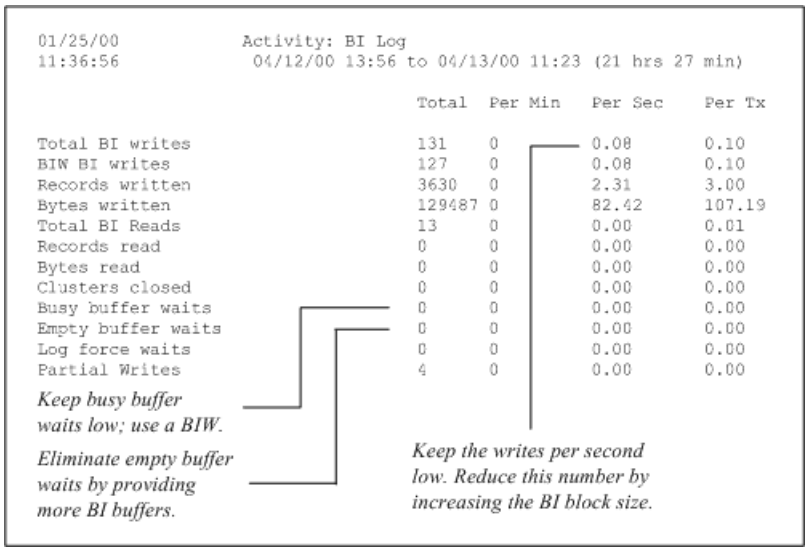
You can reduce the I/O impact of before-imaging by:

- Moving the BI file to its own disk
- Running a before-image writer (BIW) on systems with and Enterprise database license
- Providing more BI buffers
- Increasing the BI cluster size
- Increasing the BI block size
- Delaying BI writes

Monitoring BI activity

Use operating system utilities to monitor the amount of I/O activity on the disk where the BI files reside. Use the PROMON utility to monitor specific BI activity. Use the R&D option **BI Log Activity**. The following figure shows a sample display.

Figure 36: PROMON BI Log Activity display



Look for the following potential problems:

- Busy buffer waits.
- Empty buffer waits.
- High number of writes per second.
- High number of partial writes. A *partial write* occurs when the database engine must write data to the BI file before the BI buffer is full. This can happen if:

- An APW attempts to write a database block whose changes are recorded in a BI buffer that has not been written. Because BI notes must be flushed before the AI note is flushed, the APW writes the data in the BI buffer before the buffer is full so it can perform the AI write.
- An after-image writer (AIW) runs ahead of the BIW. Because BI notes must be flushed before the AI notes can be written, the AIW writes the BI buffer before it is full so it can perform the AI write.
- The Suppress BI File Write (`-Mf`) parameter's timer expires before the buffer is filled.

Moving the BI file

Disk I/O on page 277 explains the performance benefits of distributing database files across multiple disks. You help balance the before-image I/O load by placing the BI extents on a separate disk.

Using a before-image writer

The BIW is a background process that continually writes filled BI buffers to disk. Since writes to the BI file occur in the background, client and server processes rarely have to wait for a filled buffer to be written to disk. BIWs are optional, but highly recommended for improving I/O performance.

The server writes current information to the BI file through the current output buffer. When this buffer fills, the server places the buffer on the filled chain. The server then takes a new buffer from the empty chain and uses it as the current output buffer. If no empty buffers are available, the process must wait while a filled buffer is written to disk.

The BIW writes the filled buffers to disk and places them on the empty chain. By clearing out the filled chain, the BIW ensures that a supply of empty buffers is available to client and server processes.

You can only run one BIW per database. You must manually start the BIW, but you can start and stop the BIW process at any time without shutting down the database. See Starting Up and Shutting Down on page 73 for instructions on starting and stopping a BIW.

Providing more BI buffers

You can increase the number of before-image buffers in the before-image buffer pool with the Before-image Buffers (`-bibufs`) startup parameter. Increasing the number of buffers increases the availability of empty buffers to client and server processes. In general, initially set this parameter to 20. Increase it if there are any empty buffer waits in the PROMON **Activity** screen or in the R&D **BI Log Activity** screen.

Increasing the BI cluster size

The BI file is organized into clusters on disk. As the database engine writes data to the BI file, these clusters fill up. When a cluster fills, the engine must ensure that all modified database buffer blocks referenced by notes in that cluster are written to disk. This is known as a *checkpoint*. Checkpointing reduces recovery time and lets the engine reuse BI disk space. Raising the BI cluster size increases the interval between checkpoints.

Raising the BI cluster size can reduce the I/O overhead of writing modified database buffers to disk. It also lets you defer writes and collect more changes before writing a block; this lets you write multiple changes with the same write.

Larger cluster sizes generally increase performance. However, they also have significant drawbacks:

- Increased disk space usage for the BI file
- Longer crash recovery periods
- Longer checkpoint times (run APWs to eliminate this drawback)

To change the cluster size:

1. Use the PROSHUT command or the PROMON **Shutdown a Database** option to shut down the database.
2. Enter the following command:`proutil db-name -C truncate bi -bi size`

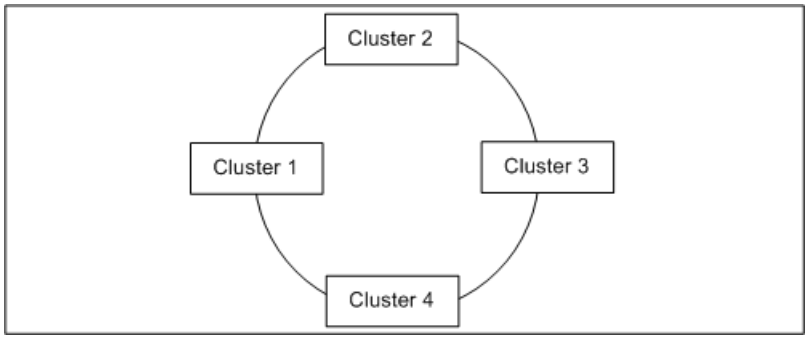
   For *size*, specify the new cluster size in kilobytes. The number must be a multiple of 16 in the range 16 to 262128 (16K-256MB). The default cluster size is 512K. Cluster sizes from 512 to 16384 are common.

   You can also change the BI block size with this command. You might want to do so at this time. For more information, see Increasing the BI block size on page 291.

Increasing the number of BI clusters

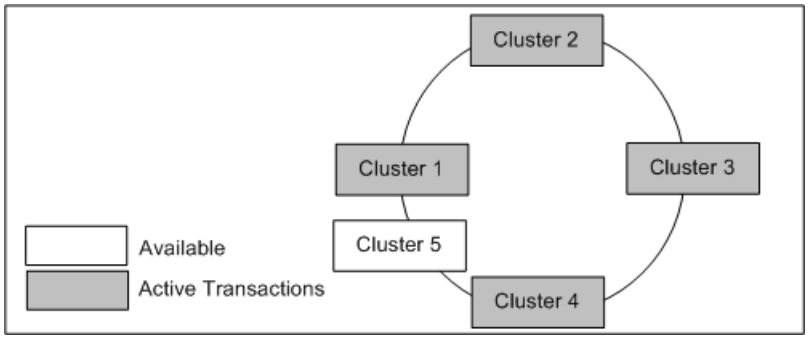
When you create a new database or truncate an existing database, the database engine, by default, creates four BI clusters, each of which is 512K. As the engine fills a cluster, the cluster is checkpointed, and the engine writes to the next cluster on the chain. The following figure illustrates the default BI clusters.

Figure 37: BI clusters at startup



In some cases, the database engine cannot write to the next cluster because the next cluster contains an active transaction. When the engine cannot use the next cluster on the chain, it creates a new cluster and begins writing to it. While the engine creates the new cluster, no database update activity can occur, thus impacting database performance. The following figure illustrates how BI clusters fill over time.

Figure 38: BI clusters over time



The BI clusters typically grow to their optimal number over time. You can calculate the current number of BI clusters for a database by dividing the BI physical file size by the BI cluster size. For example, a database BI file with a BI cluster size of 128K and a physical size of 91,7504 has 7 BI clusters.

Whenever the BI file is truncated, you should consider growing the number of BI clusters to its optimal size before restarting the database, thus preventing the database engine from adding clusters on an as-needed basis. The BI file is truncated in one of the following ways:

- Automatically by the database engine when you start after-imaging (RFUTIL AIMAGE BEGIN)
- Automatically by the database engine when you perform an index rebuild (PROUTIL IDXBUILD)
- Manually (PROUTIL TRUNCATE BI)

To increase the number of BI clusters, enter the following command:

```
proutil db-name -C bigrow n
```

For *n*, specify the number of BI clusters that you want to create for the specified database.

Increasing the BI block size

The database engine reads and writes information to the BI file in blocks. Increasing the size of these blocks allows the engine to read and write more data at one time. This can reduce I/O rates on disks where the BI files are located.

The default BI block size (8K) is sufficient for applications with low transaction rates. However, if performance monitoring indicates that BI writes are a performance bottleneck and your platform's I/O subsystem can take advantage of larger writes, increasing the BI block size might improve performance.

To change the BI block size:

1. Use the PROSHUT command or the PROMON **Shutdown a Database** option to shut down the database.
2. To change the BI block size, use the PROUTIL TRUNCATE BI command, as shown: command:

   ```
   proutil db-name -C truncate bi -biblocksize size
   ```

   For *size*, specify the new BI block size in kilobytes. Valid values are 0, 1, 2, 4, 8, and 16.

   You can also change the BI cluster size with this command. You might want to do so at this time. For more information, see Increasing the BI cluster size on page 289.

   For detailed information on this command, see PROUTIL Utility on page 629.

Delaying BI writes

When the Delayed BI File Write (`-Mf`) startup parameter is set to zero, use the Group Commit technique to increase performance. This technique assumes that for the benefit of overall performance, each individual transaction can take slightly longer. For example, when a transaction begins to commit and spools its end note to the BI buffer, it waits a short time until one of two things happen: it fills the buffer and is written to disk, or a few other transactions complete and store their end notes in the BI buffer so that a single synchronous write commits all the transactions. Use the Group Delay (`-groupdelay`) startup parameter to set the amount of time (milliseconds) the transaction waits.

If the Group Commit technique does not provide sufficient improvement, you can improve performance on a busy system by delaying BI file writes with the Delayed BI File Write (`-Mf`) startup parameter.

By default, the database engine writes the last BI block to disk at the end of each transaction. This write guarantees that the completed transaction is recorded permanently in the database. On a system with little update activity, this extra BI write is very important and adds no performance overhead. On a busy system, however, the BI write is less important (the BI block will be written to disk very soon anyway) and might incur a significant performance penalty.

Set the `-Mf` parameter to delay BI writes at transaction commit time. When `-Mf` is set to a positive value, the last BI record is guaranteed to be written to disk within the specified number of seconds. The record is written sooner if the user logs out or the system shuts down.

Note: Suppressing the last BI write does not reduce database integrity. However, if there is a system failure, the last few completed transactions can be lost (never actually written to the BI file).

For more detailed information on the `-Mf` parameter, see Database Startup Parameters on page 473.

Setting a BI threshold

When an application performs large schema updates or large transactions, the BI clusters can grow in excess of 2GB. If a crash occurs during such an operation, the recovery process might require several times the amount of disk space as the BI log was using at the time of the crash. Often this space is not available, leaving the database in an unusable state.

Using the Recovery Log Threshold (`-bithold`) startup parameter sets the maximum size to which BI files can grow. Once the threshold is reached, the database performs an emergency shutdown. This mechanism ensures that there will be enough disk space to perform database recovery. All messages associated with the threshold are logged in the database log (`.lg`) file. These messages include:

- Value of the threshold
- Warning message if the threshold is set above 1000MB
- Warning message when recovery log files are extended
- Message that a database shutdown is occurring because the threshold has been reached

The recommended range is to set `-bithold` between three and one hundred percent (3-100%) of the largest possible recovery log file size, rounded to the nearest cluster boundary. If the threshold is set above 1000MB, the database engine issues a warning message to the display and the database log (.lg) file. The system will check the total amount of BI clusters in use each time a new cluster is marked as used. If the No Crash Protection (`-i`) is set, the recovery log threshold parameter is set to the default (none) and cannot be overridden.

Enabling threshold stall

Often a database administrator does not want the database to perform an emergency shutdown when the Recovery Log Threshold limit is reached. The Threshold Stall (`-bistall`) startup parameter quiets the database when the recovery log threshold is reached. Instead of an emergency shutdown, the database stalls forward processing until the database administrator intervenes. This provides the database administrator the options of shutting down the database, making more disk space available, and increasing the threshold amount. A message is added to the database log (`.lg`) file stating that the threshold stall is enabled.

Using PROQUIET to adjust the BI threshold

You can adjust the value of the threshold by providing a valid threshold value for the PROQUIET command on systems with an Enterprise database license. The value can be increased above the current value or reduced to a value of one cluster larger than the recovery log file size at the time the PROQUIET command is issued.

To adjust the BI threshold:

1. Use the PROQUIET command to enable a database quiet point:

```
proquiet db-name enable
```

   db-name is the name of the database for which you want to adjust the BI threshold.

Note: For more information on, and complete syntax for, the PROQUIET command, see Startup and Shutdown Commands on page 461.

During a database quiet processing point, all file write activity to the database is stopped. Any processes that attempt to start a transaction while the quiet point is enabled must wait until you disable the database quiet processing point.

2. Adjust the threshold size using the `-bithreshold` parameter:

```
proquiet db-name -bithreshold n
```

db-name specifies the name of the database for which you want to adjust the BI threshold.

n specifies the new value for the threshold.

3. Use the PROQUIET command to disable the quiet point:

```
proquiet db-name disable
```

For more information on, and the complete syntax for, PROQUIET, see Startup and Shutdown Commands on page 461.

After-image I/O

After-imaging is an optional recovery mechanism that lets you recover data and transactions if a disk fails. AI files must be kept on separate disks from the database and BI files, so after-imaging I/O activity does not contribute to I/O activity on the disks where BI and database files are stored. However, after-imaging creates a significant amount of I/O that can affect performance. You can reduce the I/O impact of after-imaging by:

- Using an after-image writer (AIW) on systems with an Enterprise database license
- Raising the AI block size

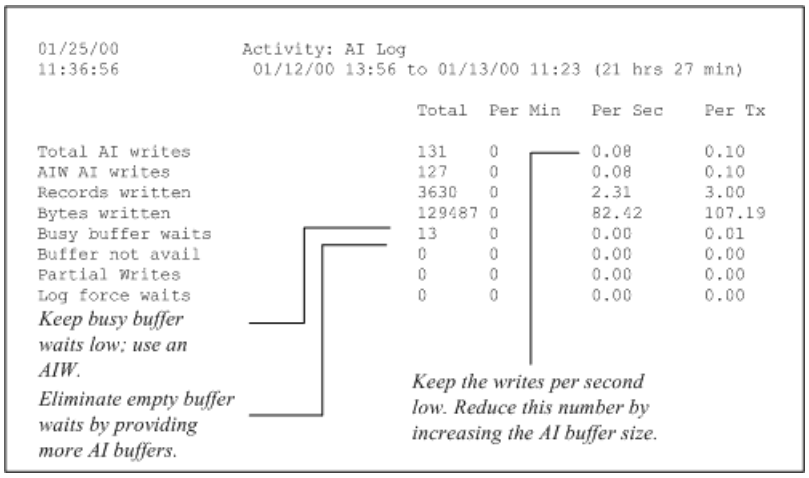
The sections that follow describe these options.

Monitoring AI activity

Use operating system utilities to monitor the amount of I/O activity on the disk where the AI files reside.

Use the PROMON utility to monitor specific AI activity. Use the R&D option **AI Log Activity**. The following figure shows a sample display.

Figure 39: PROMON AI Log Activity display



Using an after-image writer

The AIW is a background process that writes AI buffers to disk soon after they are filled. If the AIW is working effectively, client and server processes rarely have to wait for a modified buffer to be written to disk.

The AI buffer pool is a circular chain of buffers. The database engine fills these buffers one at a time. The buffer that the engine fills at any time is the current output buffer. As each buffer is filled, the engine continues around the chain, and each buffer in turn becomes the current output buffer. If the next buffer to write is already modified, the engine must wait while that buffer is written to disk.

You can run only one AIW process per database at a time. You must manually start the AIW, but you can start and stop an AIW at any time without shutting down the database. See Starting Up and Shutting Down on page 73 for instructions on starting and stopping an AIW.

Increasing the `-aibufs` startup parameter increases the number of buffers in the after-image buffer pool, which increases the availability of empty buffers to client and server processes. Set the `-aibufs` parameter to 1.5 times the value of the Before-image Buffers (`-bibufs`) parameter. (For information on setting the `-bibufs` parameter, see Providing more BI buffers on page 289.) Increasing `-aibufs` has no effect if the AIW is not running.

Increasing the AI block size

As with before-imaging, the database engine reads and writes information to the AI file in blocks. Increasing the size of AI blocks lets the engine read and write more AI data at one time. This can reduce I/O rates on disks where the AI files are located. In general, the default AI block size (8K) is sufficient for systems with low transaction rates. However, if performance monitoring indicates that AI writes are a performance bottleneck and your platform's I/O subsystem can take advantage of larger writes, increasing the AI block size might improve performance. A larger AI block size might also improve performance for roll-forward recovery processing.

To change the AI block size:

1. Use the PROSHUT command or the PROMON **Shutdown a Database** option to shut down the database.
2. If after-imaging is enabled, disable it by entering the following command:

```
rfutil db-name -C aimage end
```

For more specific information on this command, see the description of the RFUTIL utility AIMAGE END qualifier in RFUTIL Utility on page 797.

3. Truncate the BI file to bring the database and BI files up to date and eliminate any need for database recovery. To do this, enter the following command:

```
proutil db-name -C truncate bi- [ -bi size | -biblocksize size]
```

Typically, if you change the AI block size, you should also change the BI block size. If you have not already, you might want to use this command to do so. For more information on the BI block size, see Increasing the BI block size on page 291.

4. Change the AI block size by entering the following command:

```
rfutil db-name -C aimage truncate -aiblocksize size [ -a aifilename]
```

For *size*, specify the size of the AI read and write block in kilobytes. The minimum value allowed is the size of the database block. Valid values are 0, 1, 2, 4, 8, and 16. If you specify 0, RFUTIL uses the default size (8K) for your operating system platform.

5. Perform a full backup of the database.

Note: You must perform this step because backups and AI files created before you change the AI block size are no longer valid with the new AI block size. For detailed backup instructions, see Backing Up a Database on page 97.

6. Enable after-imaging by entering the following command:

```
rfutil db-name -C aimage begin { buffered | unbuffered } -a ai-name
```

For more specific information on this command, see RFUTIL Utility on page 797.

7. Restart the database and continue processing.

Direct I/O

The database engine can use an I/O technique that forces blocks to be written directly from the buffer pool to disk. This optional technique prevents writes to disk from being deferred by the operating system's buffer manager.

In general, use Direct I/O only if you are experiencing memory shortages. In many cases the normal buffered I/O will provide better performance. Test the performance impact before implementing Direct I/O on a production database.

To use this feature, specify the Direct I/O (`-directio`) startup parameter. If you use the `-directio` startup parameter, you might need to add additional APWs to compensate for the fact that with Direct I/O, each database write requires more processing time from the APWs.

Memory usage

Many of the techniques for improving server performance involve using memory to avoid disk I/O whenever possible. In general, you spend memory to improve performance. However, if the amount of memory on your system is limited, you can overload memory resources, causing your system to page. Paging can affect performance more severely than a reasonable amount of disk I/O. You must determine the point where memory use and disk I/O is balanced to provide optimal performance. In other words, you must budget the amount of memory you can afford to spend to improve performance.

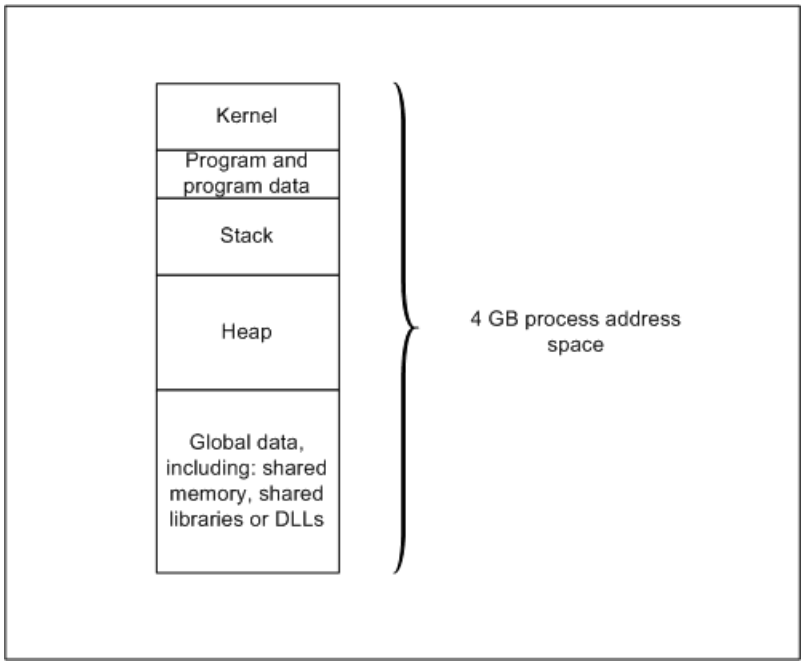
The following sections provide a memory model overview and information on monitoring and controlling memory consumption in OpenEdge:

- Process address space overview on page 296
- Controlling memory use on page 297
- Shared memory allocation on page 297
- Monitoring memory use on page 298
- Shared memory summary on page 298

Process address space overview

Many factors affect the total amount of shared memory that the OpenEdge RDBMS can allocate, including program size, stack space required, shared libraries, memory-mapped files, and memory allocated from the heap. The address space and memory allocation policies used by different operating systems are complex and vary considerably from one operating system to another. The following figure provides a sample layout for a generic 32-bit process address space. For the precise layout of your system, consult your operating system documentation.

Figure 40: Sample 4GB process address space



Most 32-bit operating systems divide a 4 GB address space into different address ranges or sections. Each section has explicit purposes. In the previous figure, the 4 GB process address space is divided into multiple sections including, the kernel, the program and its data, stack space, heap space used for dynamic memory allocation, and global data that includes shared memory and shared libraries or `.dll` files. The size of each section varies depending on operating system and does affect the size of allocated shared memory segments.

For 64-bit operating systems, most computational limits are theoretical and constrained by the available hardware. See your operating system documentation for specific information.

Controlling memory use

The table below lists the startup parameters used to fine-tune memory allocation on the server system.

Table 46: Startup parameters that affect memory allocation

Startup parameter	Suggested use
Blocks in Database Buffers (`-B`)	Increasing the buffer size decreases the amount of database record I/O by increasing the number of buffers available in memory. This increases memory usage. Increase the `-B` parameter to use more memory to reduce I/O. Decrease the `-B` parameter if memory is limited or if database buffer I/O causes paging.
Maximum Clients per Server (`-Ma`)[5]	If some number of remote clients overloads the server or exhausts the file descriptors on a system, set this parameter to limit the number of clients.
Maximum Servers (`-Mn`)[5]	If a server becomes overloaded with clients, set this parameter to limit the number of servers. If you significantly increase this parameter, you should also increase the Minimum Clients per Server (`-Mi`) parameter.
Number of Users (`-n`)	Set this parameter large enough to include both local and remote users.
Pin Shared Memory (`-pinshm`)	Use this parameter to prevent the database engine from swapping shared memory contents to disk.

Shared memory allocation

Shared memory segment allocation methodologies are dynamic; the optimum size of shared memory segments is dynamically calculated at runtime. The broker attempts to allocate the largest possible segment size and the fewest number of segments to meet the shared memory requirements.The result is that larger amounts of shared memory can be allocated more efficiently. An added benefit is that you can make more self-service database connections with large shared memory.

The **maximum** size of shared memory segments can be specified at startup. Increasing the size of shared memory segments decreases the number of segments allocated. In Windows, memory mapped files are used instead of segments. Shared memory size is specified with the `-shmsegsizen` server parameter. See Shared memory segment size (-shmsegsize) on page 529 for a description of the parameter and its arguments.

The amount of shared memory allocated for the database is constrained by your system as follows:

[5] Relevant only to database accessed using client/server.

- The database server cannot create a shared memory segment larger than the operating system maximum shared memory segment size. If the value specified for `-shmsegsize` is larger than the operating system maximum, the operating system maximum will be used. Not all platforms specify a maximum shared memory segment size; for example, Windows and AIX do not. For many of the other supported UNIX platforms, the maximum shared memory segment size is determined by the kernel parameter `SHMMAX`.
- The database server cannot create more shared memory segments than the maximum allowed by the operating system. This value is not tunable on all systems, but for many of the supported UNIX platforms, the kernel parameter for this maximum is `SHMSEG`.
- The database server cannot create shared memory segments that exceed the maximum addressable memory of a system. For 32-bit platforms, the theoretical maximum is 4 GB, but in practice, the maximum is smaller, as other aspects of the process (heap, stack, code, etc.) consume part of the available memory. For 64-bit platforms, physical system limitations keep the practical maximum well below any theoretical computational maximum. If the amount of shared memory requested exceeds the capacity of the system, the server will not start.

See your operating system documentation for specific information on shared memory segment settings.

Monitoring memory use

Use the following PROMON options to monitor memory usage:

- **Activity** — Shows the amount of shared memory used by the database and the number of shared-memory segments
- **Shared Resources Status (an R&D option)** — Shows the amount of shared memory allocated
- **Shared-memory Segments Status (an R&D option)** — Shows the ID number, size, and amount used for each shared-memory segment

For detailed information on these options, see PROMON R&D Advanced Options on page 565.

Shared memory summary

The shared memory addressing enhancements increase the shared memory addressability and increases the flexibility of shared memory allocation when the `-shmsegsize` configuration parameter is used. On 32-bit platforms, the OpenEdge RDBMS will allocate more than 2 GB of shared memory if the operating systems and machine supports the increased shared memory allocations. Even though the OpenEdge RDBMS can address larger shared memory pools than it did in prior releases, it can still only allocate and utilize as much shared memory as the operating system allows. Understanding the basic principles used by your operating system to "build" a process address space will help you understand how to set your configuration parameters when you start your databases.

Alternate Buffer Pool

Enterprise Database licenses allow for the allocation of an Alternate Buffer Pool. The Alternate Buffer Pool is a set of buffers in shared memory that are logically separate from the primary buffer pool. Management of the buffers in the Alternate Buffer Pool is independent of the primary buffer pool. Assigning specific database areas or objects to occupy buffers from the Alternate Buffer Pool might improve your buffer hit rate, thereby reducing the need to read and write buffers to and from disk, and possibly improving performance.

- Overview on page 299
- Designating objects to the Alternate Buffer Pool on page 299

- Viewing Alternate Buffer Pool selections on page 301
- Allocating Alternate Buffer Pool on page 303

Overview

In certain deployments, the buffer pool is a bottleneck that negatively impacts overall database performance due to low buffer hit ratios. The traditional solution to low buffer hit ratios is to increase the size of the buffer pool (`-B`), but this is not always possible or beneficial. The Alternate Buffer Pool gives the database administrator the ability to modify buffer pool behavior by designating objects or areas to consume buffers from the Alternate Buffer Pool, rather than from the primary buffer pool. The size of the Alternate Buffer Pool is specified with the startup parameter `-B2`. Both the Alternate Buffer Pool and the primary buffer pool consume shared memory, and the sum total of both buffer pools is limited by the established shared maximums.

The management of buffers in the Alternate Buffer Pool is separate from the primary buffer pool. As buffers in the Alternate pool are initially consumed, a First In-First Out (FIFO) queue of the buffers is maintained. When all the buffer are consumed, a Least Recently Used (LRU) replacement algorithm is employed to select buffers for eviction. The first buffer evicted is the head of the FIFO queue. However, the goal is to size the Alternate Buffer Pool such that there are always free buffers and the LRU replacement is not necessary. Monitoring of the Alternate Buffer Pool with PROMON indicates if the LRU replacement is activated. If possible, the database administrator should increase the size of the Alternate Buffer Pool with PROUTIL INCREASETO, or the database administrator should consider removing objects from the Alternate Buffer Pool to avoid utilizing LRU replacement. If the size of the Alternate Buffer Pool is increased, the LRU replacement algorithm is once again deactivated.

If you change buffer pool assignments for objects at runtime, existing buffers remain in the buffer pool where they were originally allocated.

Private read-only buffers (`-Bp`) are always obtained from the primary buffer pool regardless of the buffer pool designation of the object.

Database control and recovery areas cannot be assigned to the Alternate Buffer Pool.

If you do not specify a size for the Alternate Buffer Pool at startup, all objects and areas consume buffers from the primary buffer pool.

Specifying the best objects for the Alternate Buffer Pool is application-specific. Tables considered "hot" (very active) are good candidates, as are their related indexes, provided that the alternate buffer pool is sized to avoid requiring the use of the LRU replacement algorithm. Tables and indexes that are governed by an encryption policy are also considered good candidates because the cost of encrypting and decrypting blocks as they are written and read from disk can be high.

Designating objects to the Alternate Buffer Pool

You must identify the objects or areas that are assigned to the Alternate Buffer Pool. For areas, the PROUTIL ENABLEB2 command handles the assignment; for tables, indexes, and LOBs, the **Alternate Buffer Pool Maintenance** in Data Administration or new SQL syntax handles the assignments. Both scenarios are discussed in greater detail in the following sections:

- Specifying areas for the Alternate Buffer Pool on page 300
- Specifying objects for the Alternate Buffer Pool on page 300

Specifying areas for the Alternate Buffer Pool

Areas are designated for the Alternate Buffer Pool with the PROUTIL ENABLEB2 command, as shown:

```
proutil db-name -C enableB2 area-name
```

This command takes as input both Type I and Type II areas. The database must be offline when the command is run.

Specifying objects for the Alternate Buffer Pool

You can identify objects for the Alternate Buffer Pool with **Alternate Buffer Pool Maintenance** in **Data Administration**. The objects must reside in a Type II area to be individually designated for the Alternate Buffer Pool. Object-level assignments can be made while the database is online or while the database is in single-user mode.

For ABL objects, use **Data Administration**, and choose **Admin** > **Alternate Buffer Pool** > **Alternate Buffer Pool Maintenance**. Follow the progression of dialog boxes to select and deselect objects and designate objects for the Alternate or primary buffer pools. For details, see the Data Administration online Help, or *OpenEdge Development: Basic Database Tools*.

For SQL objects, you can designate objects to the Alternate Buffer Pool with the CREATE/ALTER TABLE and CREATEALTER INDEX statements. For details on these statements, see *OpenEdge Data Management: SQL Development*.

Data definition files

Dumping and loading data definition (`.df`) files preserves buffer pool designations. The key word, BUFFER-POOL is present from Release 10.2B forward. Clients earlier than 10.2B will not be able to load a 10.2B or higher data definition file with an object assigned to the Alternate Buffer Pool unless you choose to commit with errors. The data definition trailer also contains a `bufpool=yes` entry to signal to the load process that the file contains Alternate Buffer Pool settings. The following excerpt from a data definition file shows the new keywords:

```
UPDATE TABLE "Customer"
  BUFFER-POOL "Alternate"

UPDATE INDEX "Comments" OF "Customer"
  BUFFER-POOL "Alternate"

UPDATE INDEX "Country-Post" OF "Customer"
  BUFFER-POOL "Alternate"

UPDATE INDEX "Cust-Num" OF "Customer"
  BUFFER-POOL "Alternate"

UPDATE INDEX "Name" OF "Customer"
  BUFFER-POOL "Alternate"

UPDATE INDEX "Sales-Rep" OF "Customer"
  BUFFER-POOL "Alternate"

UPDATE TABLE "Order"
  ENCRYPTION YES
  CIPHER-NAME AES_CBC_128

UPDATE TABLE "Order-Line"
  ENCRYPTION YES
```

```
  CIPHER-NAME AES_CBC_128
  BUFFER-POOL "Alternate"

.PSC
encpolicy=yes
bufpool=yes
cpstream=ISO8859-1

.0000018384
```

Viewing Alternate Buffer Pool selections

You can view your Alternate Buffer Pool selections in two ways, as discussed in the following sections:

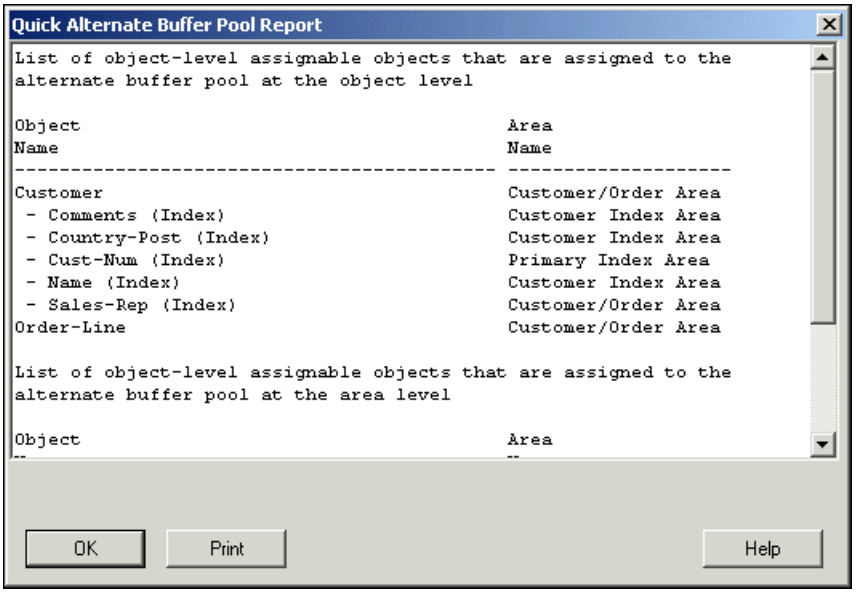
- Viewing Alternate Buffer Pool settings with the Data Dictionary on page 301
- Viewing Alternate Buffer Pool settings with PROUTIL on page 301

Viewing Alternate Buffer Pool settings with the Data Dictionary

The Data Dictionary provides a report on objects that are visible to ABL and assigned to the Alternate Buffer Pool through a specific object-level assignment. The dictionary does not report on area-level assignments or SQL-specific objects.

To view the Alternate Buffer Pool report, select **Database** > **Reports** > **Alternate Buffer Pool** from **Data Administration**.

The following report appears:



To view area-level assignments, or to see objects in the primary buffer pool, you must use PROUTIL.

Viewing Alternate Buffer Pool settings with PROUTIL

PROUTIL VIEWB2 provides a report on all area-level and object-level assignments to both the alternate and primary buffer pool.

- To view the buffer pool report, enter the following command:

```
proutil db-name -C viewB2
```

  For each area, the area level assignment and each object in the area is displayed. For each object, Object Enablement contains either Default or Alternate. Default indicates that the object takes its buffer pool assignment from the area. Alternate indicates the object is assigned to the Alternate Buffer Pool, regardless of the area assignment. An object level assignment of Alternate is only available for objects in Type II storage areas.

  Index and LOB objects are described by their object name followed by their associate object's name (table name.) For example, the `cust-num` index associated with the `customer` table in the `PUB` schema is described as `cust-num (PUB.customer)`.

The following is an excerpt of VIEWB2 output:

```
OpenEdge Release 10.2B1P as of Wed Oct 21 19:01:48 EDT 2009
Buffer pool assignment for areas and objects:
.
.
.
Area  8: "Customer/Order Area" - Alternate Buffer Pool

Object Enablement   Type  Object Id: Name
-----------------  ------ ---------------
Alternate           Table       2:  PUB.Customer
Default             Table       4:  PUB.Order
Alternate           Table       5:  PUB.Order-Line
Alternate           Index      16:  Sales-Rep (PUB.Customer)
Default             Index      21:  Cust-Order (PUB.Order)

Area  9: "Primary Index Area" - Primary Buffer Pool

Object Enablement   Type  Object Id: Name
-----------------  ------ ---------------
Alternate           Index      12:  Cust-Num (PUB.Customer)
Default             Index      20:  Order-Num (PUB.Order)
Default             Index      24:  order-line (PUB.Order-Line)

Area  10: "Customer Index Area" - Primary Buffer Pool

Object Enablement   Type  Object Id: Name
-----------------  ------ ---------------
Alternate           Index      13:  Comments (PUB.Customer)
Alternate           Index      14:  Country-Post (PUB.Customer)
Alternate           Index      15:  Name (PUB.Customer)
```

In the excerpt, observe that:

- Area 8 is assigned to the Alternate Buffer Pool at the area level. In addition, the `Customer` and `Order-Line` tables, and the `Sales-Rep` index, are explicitly assigned at the object level (via the Data Dictionary). In this example, all objects in Area 8 are assigned to the Alternate Buffer Pool due to the area level buffer pool assignment.
- Area 9 is assigned to the primary buffer pool, but its index object `Cust-Num` is assigned to the Alternate pool.

Allocating Alternate Buffer Pool

Once you have identified objects or areas for the Alternate Buffer Pool, you must allocate shared memory for the pool. Alternate Buffer Pool memory is identified by the `-B2` startup parameter. Shared memory maximums apply to the sum of your `-B2` and `-B` values. The optimum value depends on your application. You cannot specify blocks in Alternate Buffer Pool for a Replication target database. See Blocks in Alternate Buffer Pool (-B2) on page 486 for details on specifying `-B2`.

You can monitor the Alternate Buffer Pool with PROMON or through VSTs. You can modify the amount of memory allocated to the Alternate Buffer Pool online with PROUTIL INCREASETO. See the Database Administration online help or PROUTIL INCREASETO qualifier on page 736 for details on using PROUTIL INCREASETO.

Operating system resources

The database engine relies on operating system resources for its performance; for example, the engine uses the operating system file system and processes to perform its functions. These mechanisms are controlled by kernel parameters. The following sections describe these mechanisms and the kernel parameters associated with them.

Processes

The following functions run as processes:

- Brokers
- Servers
- Clients
- APWs, BIWs, and AIWs

The user table contains one entry per process. Use the Number of Users (`-n`) parameter to specify the number of users.

On UNIX, the `NPROC` parameter limits the total number of active processes on the system and is commonly set between 50 and 200. The `MAXUP` parameter limits the number of concurrent processes that can be started by a single user ID, and it is commonly set between 15 and 25. Also, if more than one user logs in with the same user ID, `MAXUP` can be exceeded quickly. The following error message is displayed when the process limit is exceeded:

```
Unable to fork process.
```

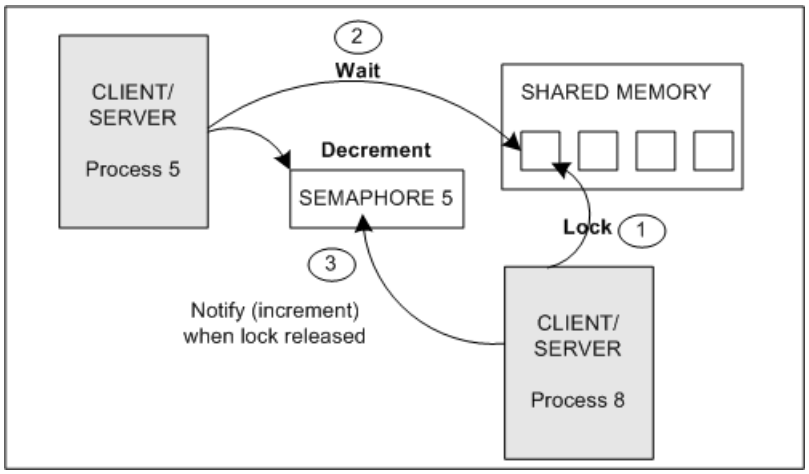
If you see this message repeatedly, you should reconfigure your system kernel.

Semaphores

On single-processor systems, semaphores are used to synchronize the activities of server and self-service client processes that are connected to a database. By default, each database has an array of semaphores, one for each user or server. Each process uses its semaphore when it must wait for a shared resource. Semaphores are not used for single-user sessions or for client sessions connecting to a remote database on a server system.

The following figure shows how semaphores control access to shared resources.

Figure 41: How a semaphore coordinates concurrent access



When process 5 needs access to a record, index, or other shared resource already locked by process 8, process 5 decrements its semaphore. When the process holding the lock (process 8) releases the resource, it notifies the waiting process (process 5) by incrementing the semaphore.

Semaphores are grouped into *semaphore sets*. Each semaphore set has a unique identifying number called a *semid*. Within a semaphore set, individual semaphores are identified by an integer ranging from 0 to one less than the size of the semaphore set.

The OpenEdge broker preallocates semaphores when you start the database with PROSERVE. Each process requires one semaphore. The broker uses two additional semaphores internally. The database engine uses the following formula to determine the number of semaphores (#SEM) to allocate:

```
#SEM = Max-possible-users (-n) + Max-possible-servers (-Mn) + 4
```

The table below lists the UNIX kernel parameters that control the number and size of the semaphore sets.

Table 47: UNIX kernel parameters that affect semaphores

Parameter	Description	Recommended setting
SEMMNI	The maximum number of semaphore identifiers allowed for the system	One per active multi-user database. If you set this value too low, the database engine might generate error 1131.
SEMMSL	The maximum number of semaphores allowed per semaphore identifier (semaphore set)	(Max-local-users-on-any-databases + Max-#servers-on-any-databases + 4). If you set this value too low, the database engine might generate error 1093 or 1130.

SEMMNS	Total number of semaphores allowed for the system	SEMMSL * number of active databases. If you set this value too low, the database engine might generate error 1093, 1131, or 1195.
SEMMNU	Maximum number of semaphore undo structures allowed for the system	Same value as SEMMNS. If you set this value too low, the database engine might generate error 1081.

When you install the OpenEdge RDBMS, you might have to increase the values of these parameters. If you are running other software that uses semaphores, take into account the combined requirements. See your system documentation for information on how to change these parameters.

The amount of kernel memory required for semaphores is relatively small, so setting the limits higher than your current needs probably will not affect performance.

The PROMON R&D **Shared Resources** option displays the number of semaphores used. When you start the broker process, a message specifies the number of semaphores still available. If the number of database users grows large, the database engine might exceed the maximum number of semaphores allowed, specified by the SEMMNS parameter. If this happens, you must reconfigure the system's kernel to increase the semaphore limit. You can reduce semaphore use only by lowering the values of the Number of Users (-n) and/or Maximum Servers (-Mn) startup parameters.

Allocating semaphores

By default, the database engine uses one semaphore set for all the semaphores needed by the database. When greater than 1000 users connect to a single database, there might be high contention for the semaphore set. Using multiple semaphore sets helps alleviate this contention and improve performance with high user counts. The broker startup parameter, Semaphore Sets (-semsets), allows you to change the number of semaphore sets available to the OpenEdge broker.

The broker uses two groups of semaphores, Login and User. The Login semaphore is used during connection to the database. The system allocates one User semaphore for every user specified by the Number of Users (-n) startup parameter. User semaphores are allocated using a round robin mechanism. If you specify the number of Semaphore Sets, PROSERVE allocates one set for the Login semaphore and the remaining sets are used for User semaphores.

In this example, the broker uses two semaphore sets, one for the Login semaphore and one for the ten User semaphores:

```
proserve db-name -semsets 2 -n 10
```

In this example, the broker uses three semaphore sets, one for the Login semaphore, one for five of the User semaphores, and one for the remaining five User semaphores:

```
proserve db-name -semsets 3 -n 10
```

Spin locks

On multi-processor systems, the database engine uses a spin lock algorithm to control access to memory. The spin lock algorithm works as follows: When a process needs a memory resource, it attempts to acquire the resource's latch. If it cannot acquire the resource's latch, it repeats the attempt. This iterative process is called *spinning*. If a process fails to acquire a latch after a specified number of spins, the process pauses, or takes a nap, before trying again. If a process repeatedly fails to acquire a latch, the length of its nap is gradually increased. You can set the Spin Lock Retries (`-spin`) parameter to specify how many times to test a lock before napping.

File descriptors

A *file descriptor* is an identifier assigned to a file when it is opened. There is a system limit on the number of file descriptors. If you have insufficient file descriptors, you might not be able to start a broker or connect a new client to an existing broker. See OpenEdge RDBMS Limits on page 59 for details on calculating the number of file descriptors required.

Network Communication

Several tuning parameters for OpenEdge databases with an Enterprise license are available to improve performance of prefetch queries in large scale deployments. A prefetch query is a NO-LOCK query against a database being accessed across a network. Prefetch queries often return a large number of records. Tuning parameters to improve the performance of these queries is discussed in the following sections:

- Prefetch tuning on page 306
- Message wait on page 307

Prefetch tuning

Tuning the behavior of prefetch queries can mitigate two possible performance issues: high System CPU utilization and inefficient network utilization.

The poll() mechanism employed in the OpenEdge client/server environment is known to have a high System CPU cost associated with it. In large-scale deployments, the cost of the poll() mechanism can overwhelm a system, preventing proper scaling. By managing the frequency of poll() calls, you can reduce System CPU utilization and increase user CPU utilization.

Additionally, using large network message buffers to make most efficient use of network bandwidth is often subverted when the remote client digests the received data faster than the server can populate a network message to its capacity. By managing the contents and behavior of network messages, you can more effectively manage your network bandwidth.

OpenEdge provides the ability to manage the contents and behavior of network messages and the frequency of calls to the poll() mechanism to mitigate these issues. Using tuning parameters, you can fine tune the behavior of message packing for prefetch queries as well as how often the poll() system call is invoked.

Enabling prefetch delay

A network message containing the first record associated with a prefetch query was immediately sent to the remote client requesting it. Subsequent messages included the remaining records. Enabling a delay through the startup parameter `-prefetchDelay`, allows the OpenEdge server to fill the network message containing the first record with additional records before sending it to the remote client. Two additional startup parameters, `-prefetchNumRecs` and `-prefetchFactor`, determine the duration of the delay.

If you specify -prefetchDelay, only the behavior of the first network message is altered. The default behavior when -prefetchDelay is not specified, is to continue sending the first record in its own message, regardless of the -prefetchNumRecs and -prefetchFactor parameter values.

Specifying number of records prefetched

The -prefetchNumRecs parameter specifies a number of records to attempt to put into a network message prior to sending the message across the network. A higher value can create a longer delay in sending the message. This delay can improve overall system performance by reducing network traffic, but individual clients may encounter choppy responses from the server. The range of allowable values for- prefetchNumRecs is between 1 and 32,766. The default is 16, which is identical to the behavior before this value was tunable.

Specifying the percent of a network buffer used for prefetch records

The -prefetchFactor parameter specifies a "percent full" calculation of a network message. The allowable range is from 0 to 100, where 0 is the equivalent of turning off this tuning parameter, and also the default.

Calculating prefetch buffer utilization

You can alter network message utilization for a prefetch query with the startup parameters -prefetchNumRecs and -prefetchFactor. NumRecs takes an integer value that represents the number of records to put in the network message. Factor takes an integer value that represents a "percent full" calculation for the network message. The network message is sent when the **minimum** of either parameter is reached. The interaction of the two parameters is governed as follows:

- When NumRecs is set extremely high, it guarantees that the Factor percentage is the minimum.
- When Factor is set to zero (0), it guarantees that the number of records specified by NumRecs is the minimum.
- By default, the size of the prefetch data in a network message is controlled by the number of records (16).

Prefetch priority

In deployments with exceptionally heavy utilization of the poll() mechanism, you can add the ability to skip calls to poll() with the startup parameter -prefetchpriority. This parameter takes an integer value that represents the number of records to add to the network message of an in-process prefetch query without additional polling. The poll skip is off by default, and only engaged when the previous poll() call returns with nothing waiting.

In some circumstances, the poll() call actually takes longer than the addition of a record to the message in the ready queue. Adding more than one record to the message at a time can greatly reduce the number of poll() calls being made on large deployments. The expected outcome is that skipping calls to poll() greatly improves overall system performance due to a decrease in "System CPU" cycles, allowing for increased "User CPU" cycles.

Message wait

In a deployment where there is heavy network traffic, you can add a message wait to the server such that the server waits the specified number of seconds to have communication with a remote client before looking for other events, such as a database shut down or force disconnect of a remote client. Setting this value can reduce system calls when a server is idle. The default is 2 (the same as the previous behavior).

For more information, see:

- Database Startup Parameters on page 473 for details on specifying these startup parameters.

- PROMON Utility on page 543 for details on monitoring and modifying these parameters while your database is online.

Database fragmentation

Over time, as records are deleted from a database and new records are added, gaps can occur on the disk where the data is stored. This *fragmentation* can cause inefficient disk space usage and poor performance with sequential reads. You can eliminate fragmentation by dumping and reloading the database. You can manage fragmentation by changing create and toss limits.

Analyzing database fragmentation

To determine the degree of fragmentation for tables in a database, use PROUTIL TABANALYS as shown:

```
proutil db-name -C tabanalys
```

You can run PROUTIL TABANALYS while the database is in use; however, PROUTIL generates only approximate information.

In the TABANALYS display, check the following fields:

- **Count** — The total number of record fragments found for each table in the database.
- **Fragments Factor** — The degree of record fragmentation for each table. If the value is 2.0 or greater, dumping and reloading will improve disk space usage and performance. If the value is less than 1.5, dumping and reloading is unnecessary.
- **Scatter Factor** — The degree of distance between records in the table. The optimal value for this field varies from database to database. To determine the optimal value for your database, run the TABANALYS on a freshly loaded database.

The following sample shows an excerpt from a PROUTIL TABANALYS display:

```
RECORD BLOCK SUMMARY FOR AREA "Info Area" : 7
-----------------------------------------------------
                               -Record Size (B)- ---Fragments--- Scatter
Table        Records      Size   Min   Max  Mean   Count  Factor     Factor
Customer          83     12.1K   118   222   149      83     1.0        1.5
Invoice          147     5613B    32    45    38     147     1.0        1.2
Item              55     5092B    39   229    92      55     1.0        1.3
Local-Default     10      704B    55    82    70      10     1.0        1.2
Ref-Call          13     2480B    81   328   190      13     1.0        1.4
Salesrep           9      746B    79    87    82       9     1.0        1.2
State             51     1755B    29    40    34      51     1.0        1.0

RECORD BLOCK SUMMARY FOR AREA "Order Area" : 8
-----------------------------------------------------
                               -Record Size (B)- ---Fragments--- Scatter
Table        Records      Size   Min   Max  Mean   Count  Factor     Factor
Order            207     11.5K    53    61    56     207     1.0        1.3
Order-Line       873     30.3K    33    38    35     873     1.0        1.2
         ------------------------------------------------------------------
Totals:         4190     431.1K     6  1156   105    4263     1.0        2.5
```

Eliminating database fragmentation

You can eliminate database fragmentation in two ways:

- Dump and reload the database.

  Dumping and reloading the database creates a new starting version of the database and loads the table contents into this new database. During the loading stage, any gaps created when database elements were deleted are removed, thereby more efficiently allocating disk space. There are many methods available to dump and load data; for instructions, see Dumping and Loading on page 411.

- Move individual tables and indexes.

  You can use PROUTIL TABLEMOVE and PROUTIL IDXMOVE to move tables and indexes from one storage area to another while the database remains online. As a result, disk space is efficiently allocated and gaps are removed. For more information about moving tables and indexes, see Maintaining Database Structure on page 325.

Managing fragmentation

Records are allocated to a block or blocks according to algorithms that aim to maximize storage utilization, and minimize fragmentation. When allocating space for a new record, the database engine first attempts to find space in the blocks on the RM (record management) chain. The RM chain contains partially filled blocks. If a block is not found on the RM chain, the record will be inserted into a block from the free chain. The free chain contains empty blocks. Changing create and toss limits determines where new records are inserted and the amount of unused space a block must contain when on the RM chain. Unused space is intentionally left within a block to allow for future expansion of the record. The goal is that the free space is sufficient to handle anticipated growth, eliminating the need to split the record across blocks.

The limits delineating the free space thresholds are defined as:

- **Create Limit** — The minimum amount of free space in bytes required in a block after a new record is inserted. The create limit must be greater than 32, and less than the block size minus 128. For a database with a 1K block size, the default create limit is 75. For all other block sizes, the default create limit is 150.
- **Toss Limit** — The minimum amount of free space in bytes required in a block for the block to remain on the RM chain. Once a block contains fewer empty bytes than the toss limit, it is removed from the RM chain, and any remaining space is dedicated to record growth within the block. For a database with a 1K block size, the default toss limit is 150. For all other block sizes, the default toss limit is 300.

The create and toss limits are changed using PROUTIL. For Type I storage areas, create and toss limits are changed on a per area basis. For Type II storage areas, limits are changed on a per area or object basis. The general syntax for changing a create or toss limit with PROUTIL is as follows:

Syntax

```
proutil db-name -C SET-obj-limittype-LIMIT object-idenitifer new-limit
```

The table below describes the PROUTIL qualifiers for changing create and toss limits.

Table 48: PROUTIL qualifiers for changing create and toss limits

Limit	Object type	Qualifier
Create	Area	PROUTIL SETAREACREATELIMIT qualifier on page 760
	BLOB	PROUTIL SETLOBCREATELIMIT qualifier on page 762
	Table	PROUTIL SETTABLECREATELIMIT qualifier on page 764
Toss	Area	PROUTIL SETAREATOSSLIMIT qualifier on page 761
	BLOB	PROUTIL SETLOBTOSSLIMIT qualifier on page 763
	Table	PROUTIL SETTABLETOSSLIMIT qualifier on page 766

The current values for create and toss limits are displayed by using PROUTIL DISPTOSSCREATELIMITS. The limits are displayed for a specified area. See PROUTIL DISPTOSSCREATELIMITS qualifier on page 688 for the complete syntax.

Create and toss limits should only be changed when your database experiences high rates of fragmentation or inefficient space utilization within blocks. The table below describes situations and suggested solutions.

Table 49: Create and toss limit situations and solutions

Situation	Suggested solution
Fragmentation occurs on updates to existing records. You anticipated one fragment, but two were created.	Increase create limit
Fragmentation occurs on updates to existing records or you have many blocks on the RM chain with insufficient space to create new records.	Increase toss limit
There is limited fragmentation, but database block space is being used inefficiently, and records are not expected to grow beyond their original size.	Decrease create limit
There is limited fragmentation, but database block space is being used inefficiently, and records are not expected to grow beyond their original size.	Decrease toss limit

Increasing the create and toss limits address record fragmentation by allowing more space for records to grow within a block before being continued in another block. Both limits identify an amount of space to reserve for existing records to grow within a block. If the create limit check fails, the new record is inserted into a different block. It remains possible that a smaller record can be inserted into the first block because failing the create limit check does not remove it from the RM chain. If the toss limit check fails, when a record is expanded, the block is removed from the RM chain so no new records can be inserted into it.

Decreasing the create and toss limits address problems with efficient block space utilization. When blocks have large amounts of empty space and record growth is not anticipated, but new records are unable to be added because the limit checks are failing, you can consider decreasing the create and toss limits. Decreasing the create limit increases the potential that a new record can be added to an existing record, and decreasing the toss limit will leave partially filled blocks on the RM chain.

Index use

As database blocks can become fragmented, index blocks can become under-utilized over time. The optimal degree of index block utilization depends on the type of database access performed. Retrieval-intensive applications generally perform better when the index blocks are close to full since the database engine has to access fewer blocks to retrieve a record. The larger the index, the greater the potential for improving performance by compacting the index. Update-intensive applications, on the other hand, perform better with loosely packed indexes because there is room to insert new keys without having to allocate more blocks. Index analysis provides the utilization information you require to make decisions. Choose a balance between tightly packed indexes and under-utilized indexes, depending on your data and applications. The sections that follow describe the following index activities:

- Analyzing index use on page 311
- Compacting indexes on page 312
- Rebuilding indexes on page 315
- Activating a single index on page 320

Analyzing index use

Use PROUTIL IDXANALYS to get information about index blocks and usage, as shown:

```
proutil db-name -C idxanalys
```

The IDXANALYS qualifier provides:

- The number of fields and levels in each index
- The size of each index, in blocks and in bytes
- The percent utilization within the index (that is, the degree of disk space efficiency)
- A factor value that indicates whether to rebuild each index
- A summary of indexes for the current database and the percentage of total index space used by each index

Note: You can run PROUTIL IDXANALYS while the database is in use; however, PROUTIL generates only approximate information.

The most important field in the IDXANALYS display is the **% Util** field. This field shows the degree of consolidation of each index. If an index is several hundred blocks and your application most frequently retrieves data, an index utilization of 85 percent or higher is optimal. There are two ways to increase an index's utilization rate:

- Compress the index with the database online or offline with the PROUTIL IDXCOMPACT utility
- Rebuild and compress the index offline with the PROUTIL IDXBUILD utility

The Levels field shows the number of reads PROUTIL performs in each index per entry. The Blocks and Bytes fields show you the size of each index. The Factor field is based on the utilization and size of the index; it is an indicator of when you should rebuild indexes. The table below provides a description of the different ranges of values for the Factor field. When you use the Factor field to decide whether to rebuild an index, consider the context of how the particular index is used. For example, if an index is highly active, with continuous insertions and deletions, its utilization rate varies greatly, and a rebuild is inadvisable. However, a static index with a high factor value benefits from a rebuild.

Table 50: Factor values

Factor range	Description
1 to 2	The index is well-utilized and balanced. You do not have to rebuild it.
2 to 2.5	The index is less than 50 percent utilized and/or the index is unbalanced. You should consider a rebuild.
2.5 to 3	The index is less than 25 percent utilized and/or the index is very unbalanced. You should rebuild this index.

Compacting indexes

When space utilization of an index is reduced to 60 percent or less as indicated by the PROUTIL IDXANALYS utility, use the PROUTIL IDXCOMPACT utility to perform index compaction online. Performing index compaction increases space utilization of the index block to the compacting percentage specified. For example:

```
proutil db-name -C idxcompact [ owner-name. ] table-name.index-name [n]
```

Note: For the complete syntax description see PROUTIL utility syntax on page 632

Performing index compaction reduces the number of blocks in the B-tree and possibly the number of B-tree levels, which improves query performance.

The index compacting utility operates in phases:

- **Phase 1** — If the index is a unique index, the delete chain is scanned and the index blocks are cleaned up by removing deleted entries
- **Phase 2** — The nonleaf levels of the B-tree are compacted, starting at the root and working toward the leaf level
- **Phase 3** — The leaf level is compacted

The _UserStatus virtual system table displays the utility's progress. For more information, see Virtual System Tables on page 863

Note: Because index compacting is performed online, other users can use the index simultaneously for read or write operation with no restrictions. Index compacting only locks one to three index blocks at a time, for a short time. This allows full concurrency.

Index utilities

The available index utilities offer the ability to check, fix and rebuild indexes.

Index check (PROUTIL IDXCHECK)

Index Check identifies possible errors, but does not ever alter index entries. IDXCHECK can be run online or offline. If an IDXCHECK operation is interrupted or crashes, it does not cause data corruption. IDXCHECK works by:

1. Checking that all index entries associated with each record can be found.
2. Verifying that each value in the index is in the associated record.
3. Undertaking a validation of all index blocks and reporting an associated error as appropriate.

IDXCHECK is designed to detect index corruption as sensitively as possible. Because this requires many reads to take place during the process, it may take significant time to complete.

Available validation options for IDXCHECK are:

1. Validate physical consistence of index blocks (on by default)
2. Validate keys for each record (on by default)
3. Validate record for each key
4. Validate key order (on by default)
5. Validate tree
6. Validate index against free cluster

You can also choose to lock tables and change the error limit.

IDXCHECK processes the validation options in the following order:

1. Options 1 and 2 are executed together.
2. Options 3 and 4 are executed together.

Note:

Option 4 detects physical corruption in addition to validating key order. Running Option 4 online may generate false positives. If errors are detected, try re-running IDXCHECK with the option to lock the tables enabled. Locking the tables eliminates false positives.

Option 5 detects issues with indexes that span the 32/64 bit boundary of your index area in a deterministic fashion, it is off by default. The issue is also detected with Option 3, but in a non-deterministic fashion.

Option 6 detects if index entries for the special schemas have been incorrectly deleted by PROUTIL MVSCH. These errors only exist if you executed PROUTIL MVSCH on your database with OpenEdge Releases between 11.6.0 and 11.6.3 (with no hotfix).

For complete details on IDXCHECK, see PROUTIL IDXCHECK qualifier on page 722.

Index fix (PROUTIL IDXFIX)

Index Fix identifies and correct errors in indexes.

Available validation options for IDXFIX are:

1. Scan records for missing index entries.
2. Scan indexes for invalid index entries.
3. Both 1 and 2 above.

IDXFIX processes the validation options in the following order:

1. Option 2 (scan indexes)
2. Option 1 (scan records)

For complete details on IDXFIX, see PROUTIL IDXFIX qualifier on page 730

Comparison of IDXCHECK and IDXFIX

The IDXCHECK and IDXFIX both check for index corruption. The main difference is that IDXFIX can repair corrupted indexes.

The two utilities have the following common attributes:

- Both utilities can run online or offline.
- Both utilities can run multiple online sessions concurrently.
- After-imaging does not need to be disabled prior to running either utility.
- Both utilities display error messages on the screen and log them in the log file. They also display and log a success or failure indication, along with the number of errors and warnings issued.
- Both utilities display warning messages. Warning messages indicate that there is a problem, but the index is still valid.

Use the following table to contrast the differences in behavior of the two utilities:

Issue	IDXCHECK action	IDXFIX action
Missing index entries	Identify and report	Identify, report, and fix by adding missing entries
Orphan index entries	Identify and report	Identify, report, and fix by deleting orphan entries

Operational comparison

IDXFIX and IDXCHECK are at parity and detect the same index corruption when IDXCHECK is run with validation options 1, 2, 3 (not on by default), and 4, and IDXFIX is run with option 3 and are run offline.

If IDXCHECK is run with the default settings, it does not detect the corruption that can be detected by options 2 and 3 of IDXFIX. If IDXFIX is run online, it does not detect the errors found by Option 4 of IDXCHECK.

Both utilities are resource intensive, and compete with other clients for database and system resources when run online. Consider limiting the scope of the indexes processed by choosing a subset of indexes with the “Some” option versus choosing all indexes with the “All” option.

Index build (PROUTIL IDXBUILD)

Index Rebuild deletes existing indexes and rebuilds them from scratch. IDXBUILD does not make any attempt to discover errors prior to deletion. Errors encountered during the index building process are reported. If there is a fatal error, the database cannot be recovered and must be restored from backup. For this reason, it is critical to perform a backup before running IDXBUILD. When the IDXBUILD process completes, the rebuilt indexes replace the old indexes. If there was any index corruption in the old indexes, it is not detected because the IDXBUILD replaces the old indexes with the newly rebuilt indexes.

IDXBUILD runs strictly offline. Multi-tenant index rebuild (MTIDXBUILD) and table partitioned index rebuild (TPIDXBUILD) can be run online. IDXBUILD is discussed in Rebuilding indexes on page 315.

For complete details on the index rebuild utilities, see:

- PROUTIL IDXBUILD qualifier on page 715
- PROUTIL MTIDXBUILD qualifier on page 743
- PROUTIL TPIDXBUILD qualifier on page 773

Rebuilding indexes

Use the IDXBUILD (Index Rebuild) qualifier of the PROUTIL utility to:

- Compress index blocks to minimize space usage
- Activate all deactivated indexes in the database
- Repair corrupted indexes in the database (index corruption is normally signaled by error messages)

Note:

When you run the Index Rebuild, the database must not be in use.

For multi-tenant databases, you can rebuild an index for a tenant or group while the database remains online. See Multi-tenant index rebuild on page 369 for details.

For table-partitioned databases, you can rebuild an index for a partition while the database remains online. See Table-partitioned index rebuild on page 388 for details.

You perform a backup of your database immediately prior to running an Index Rebuild. Should the Index Rebuild crash due to data corruption, the only method of recovery is a restore from backup.

To run PROUTIL IDXBUILD, enter the following command:

```
proutil db-name -C idxbuild
  [ all | activeindexes | inactiveindexes |
    table [owner-name.] table-name
  [ partition partition-name | composite initial] |
    area area-name |  schema schema-owner]
  [ tenant tenant-name| group group-name | shared ]
  [ -thread n ] [ -threadnum n ]
  [ -datascanthreads n ] [ -mergethreads n ]
  [ -T dir-name | -SS sort-file-directory-specification ]
  [ -TB blocksize ] [ -TM n ] [ -TMB n ]
  [ -TF n ] [ -freeTF n ] [ -B n ]
  [ -SG n ]
  [ -pfactor n ] [ -rusage]
```

For more information on each option, see the PROUTIL IDXBUILD qualifier on page 715.

When you enter this command without the `all`, `table`, `area`, or `schema` qualifiers, the following menu appears:

```
Index Rebuild Utility
=====================

Select one of the following:
All           (a/A) - Rebuild all the indexes
Some          (s/S) - Rebuild only some of the indexes
By Area       (r/R) - Rebuild indexes in selected areas
By Schema     (c/C) - Rebuild indexes  by schema owners
By Table      (t/T) - Rebuild indexes in selected tables
By Partition  (p/P) - Choose indexes in selected table partitions
By Activation (v/V) - Rebuild selected active or inactive indexes
-------------------
Multi-Tenancy (m/M) - Choose tenants or groups
-------------------
Quit     (q/Q) - Quit, do not rebuild

Enter your selection:
```

Make your selection, considering the following:

- Use the **All** option to rebuild all indexes.
- Use the **Some** option to rebuild only specific indexes.
- Use the **By Area** option to rebuild indexes specific to one or more areas.
- Use the **By Schema** option to rebuild indexes owned by one or more schema owners.
- Use the **By Table** option to rebuild indexes specific to one or more tables.
- Use the **By Partition** option to rebuild indexes specific to one or more table partitions.
- Use the **By Activation** option to select active or inactive indexes.
- Use the **Multi-Tenancy** option to refine your selection to specific tenants or groups.

After you enter a selection and you qualify those indexes you want to rebuild, the utility prompts if you have enough disk space for index sorting. If you enter yes, the utility sorts the indexes you are rebuilding, generating the indexes in order by their keys. This sorting results in a faster index rebuild and better space use in the index blocks.

To estimate whether or not you have enough free space to sort the indexes, use the following formulas:

- If you rebuild all the indexes in your database, sorting the indexes requires up to 75 percent of the total database size in free space.
- If you rebuild an individual index, sorting that index can require as much as the following amount of free space:

```
(size of one index entry) * (number of records in file) * 3
```

PROUTIL IDXBUILD rebuilds an index or set of indexes in a series of three phases:

1. The utility scans the database by area, clearing all index blocks that belong to the indexes you are rebuilding and adding those blocks to the free block list.
2. The utility scans the database by area and rebuilds all the index entries for every data record. If you chose to sort the index, the utility writes the index entries to the sort file. Otherwise, the utility writes the index entries to the appropriate index at this point.
3. The utility sorts the index entries in the sort file into groups and enters those entries into their respective entries in order, one index at a time, building a compacted index. This phase only occurs if you chose to sort the indexes.

Index Rebuild accomplishes most of its work without displaying messages, unless it encounters an error condition.

For Enterprise database licenses, index rebuild is multi-threaded by default. You can specify the maximum number of threads created using the `-threadnum` *n* parameter. If not specified, the maximum number of threads created will equal the system's number of CPUs. The actual number of threads created will not exceed the number of index groups in an area if this value is smaller than the maximum. During a multi-threaded index rebuild, separate threads are assigned the external merging of each index group during Phase 2. Once the main process has created all the threads for Phase 2, it immediately begins building the index tree for Phase 3, enabling Phase 3 to be executed in parallel with Phase 2. If an area has only one index to rebuild, the work will be executed without the use of threads.

If you do not want your index rebuild to be multi-threaded, specify `-threads 0`. This directs the index rebuild to execute in an unthreaded mode.

If the index rebuild is interrupted while rebuilding selected indexes, the list of selected indexes is retained in a file named `dbname.xb`. This `.xb` file is used when the utility is restarted. You do not have to enter the list of indexes manually if the `.xb` file exists.

Overcoming SRT size limitations

When you run the Index Rebuild utility and choose the **Sort** option, you might encounter space limitations that can cause the utility to terminate. To overcome this limitation, simply create a file that contains specifications for the directories and the amount of space per directory that you want the `SRT` file to have access to during the Index Rebuild. The file that contains the specifications must be a text file, have the same name as the database with an extension of `.srt` (*dbname*`.srt`), and reside in the same directory as the `.db` file. In addition, the contents of the file must follow these conventions:

- List the directory and the amount of space that you want to allocate to the index rebuild sort on separate lines.
- The size that you specify in the *dbname*`.srt` directory specification is the maximum (in 1024 byte units) that the file can grow. Specifying 0 for any directory indicates that you want to allow unlimited growth.
- Separate the directories from the size by at least one blank.
- Terminate the line with a slash (/) followed by end of line.
- For threaded index rebuilds, spread the directories across as many devices as possible. In threaded builds, each thread will use the next directory in the sort file, looping back to the beginning of the list, if necessary. If multiple sort files are open on the same disk, you could create significant I/O contention, reducing the performance gain of the threaded rebuild.

For example, if you want to rebuild the index for the sports database and you want the speed sort to have access to 300K of space available in the /user2/db1/first directory, 400K in the user3/junk directory, and unlimited space in the /user4/last directory, then the sports.srt looks like this on UNIX:

```
300  /user2/db1/first/
400  /user3/junk/
0    /user4/last/
```

And looks like this for Windows:

```
300  d:\temp
400  e:\temp
0    f:\temp
```

The Index Rebuild utility accesses the files in the order in which they are listed in the *dbname*.srt file. So, if you specify an amount of space that is not available, when the disk is filled, then Index Rebuild terminates and the next directory specification is not used. Thus, if a disk has only 200K of space and the *dbname*.srt specifies 300K, when the 200K is exhausted the Index Rebuild terminates. For example, if /user2/db1/first above does not get 300K of data, Index Rebuild never processes /user3/junk. In addition, if you specify a directory size of 0, any directories specified after it in the *dbname*.srt are not processed. For these reasons, you should verify that the space you specify in the *dbname*.srt file is available before running index rebuild.

The Index Rebuild utility opens the files for each of the directories before it actually starts the sort process. As a result, one of the following messages is displayed for each file:

```
Temporary sort file at pathname will use up to nK of disk space.
```

Or:

```
Temporary sort file at pathname will use the available disk space.
```

The previous message occurs even if the .srt file was not found.

When the sort completes, the following message is displayed for each file:

```
Temporary sort file at pathname used nK of disk space.
```

In some cases the message displays 0K. This simply means that the sort took place completely in memory.

If Index Rebuild does not find a *dbname*.srt file, then by default, it uses the directory supplied by either the -T parameter or the current working directory.

Maximizing index rebuild performance

To speed up index rebuild operations, do the following:

- Answer yes when prompted whether you have enough disk space for sorting.

- Increase the Speed Sort (-TB) startup parameter to 24K. (If you are very short of memory, use 16K or 8K.) This improves sort performance; however, it also uses more memory and disk space. The default is 8, and can be set as high as 64.
- Increase the Merge Number (-TM) parameter to 32 (unless memory is scarce).
- Use the Sort Grouping (-SG) parameter. A large -SG value requires more memory allocation and more file handles. To determine the amount of memory (in kilobytes) needed for each index group, add 1 to the merge number (the value of -TM) and multiply the sum by the speed sort block size (the value of -TB). Memory consumption for each index group equals (-TM + 1) * -TB.
- Change the Temporary Directory (-T) startup parameter to store the temporary files on another disk.
- Utilize threading parameters.
  - Specify -datascanthreads to specify the number of concurrent threads spawned to scan table data associated with the index being rebuilt.
  - Specify -mergethreads to specify the number of concurrent threads spawned to execute the sort and merge phases of index rebuild.

The database engine uses the following algorithm to rebuild indexes for each record:

1. Read the index key fields and store in the first available SRT file block.
2. Allocate additional SRT file blocks of the specified block size as required to hold all index keys.
3. Sort the keys in each block, then merge the keys to produce a sorted file.

A similar technique is used to sort records when there is no index to satisfy a BY clause.

A larger block size can improve index rebuild performance considerably. A larger block size means less SRT block allocation overhead and fewer quicksort operations on individual blocks.

You might have to run the application several times using different block size values to determine the optimal value. If you experience extreme memory shortages when running an OpenEdge session, try setting the block size to 1 to reduce memory consumption.

During index rebuild, try setting -TB to 31, if memory and disk space are available. If the index rebuild fails, try successively smaller values. Remember, a larger value for -TB improves sort performance but uses more memory. The -TB setting has a significant impact on the size of the SRT temporary file. The SRT file size depends on the number of session compile files, and the number and size of sort operations.

Memory usage depends on the number of sorts simultaneously occurring. The simultaneous sorts are logically equivalent to nested FOR EACH statements. You can estimate memory usage as follows, where M is the estimated memory usage:

```
M = sort-block-size *
        (number-of-simultaneous-sorts+Merge-Number(-TM) parameter)
```

Index rebuild always requires eight simultaneous sorts, so during index rebuild:

```
M = sort-block-size *(8 + (-TM)parameter)
```

Therefore, in the default case:

```
M = (2*(8+5)) = 26K
```

Reactivating unique indexes

When reactivating a unique index, IDXBUILD displays the following error message each time it encounters a duplicate index key. For example:

```
Fix RECID recid, table-name already exists with field-name value.
```

You must change the record data to eliminate duplicate keys to access all the data with this index. Use another index on the table (if one exists):

```
FOR EACH table-name USE-INDEX index-without-duplicate-keys:
UPDATE table-name.
```

Activating a single index

Indexes that are added to your database while it is online are inactive until you specifically initiate activation with PROUTIL IDXACTIVATE. When you run IDXACTIVATE you provide the name of one inactive index on the command line. The general syntax is as follows:

```
proutil db-name -C idxactivate [owner-name.] table-name.index-name
          [useindex index-name] [recs n] [refresh t]
```

For details of the command, see PROUTIL IDXACTIVATE qualifier on page 710.

Before activating the index, IDXACTIVATE checks to make sure that there are no users with a schema timestamp that is earlier than the schema timestamp of the index. If any such users are connected to the database, IDXACTIVATE cannot immediately proceed, but does one of the following:

- In OpenEdge Release 11.7 and higher, if you have enabled Database Client Notification (`-usernotifytime`), IDXACTIVATE will wait until all the connected users respond to the notification, and then proceed with the activation.
- In releases prior to 11.7, or if Database Client Notification (`-usernotifytime`) is off (set to zero), you are given the option of waiting or canceling the index activation. If you chose to wait, you must wait until all the users with an earlier schema timestamp disconnect, or SQL clients start a new transaction, or you can use PROSHUT to forcibly remove the connections. You control the update of the status display of blocking users with the `refresh` option. The number supplied to `refresh` indicates the number of seconds between displays of blocking users.

When IDXACTIVATE activates and builds the index, it bundles records into transactions. By default, 100 records are bundled into one transaction. You can alter that number with the `recs` option. The number you specify for the `recs` option is the number of records to bundle into one transaction.

The following output shows the output of IDXACTIVATE, without Database Client Notification enabled. In this example the index `tst_table.inact3` is activated for the database `doc_db`; no optional parameters are specified so default values are used. The cycle of waiting for users to disconnect executes twice before all the users with old timestamps are disconnected and the index is activated, as shown:

```
$ proutil doc_db -C idxactivate tst_table.inact3
OpenEdge Release 10.1A1P as of Tue Oct 18 20:11:37 EDT 2005

Index Activate: BEGIN activation of inact3. (13258)
Index Activate: recs value: 100 refresh rate: 60. (13259)
Activating Index inact3  at Database timestamp 1129752317 . (12811)
  Usr      Name     Type      Pid Timestamp

        5  docusr1 Usr         10336 1129580580
        6  docusr2 Usr         11208 1129580580
Connections with timestamps older than 1129752317  exist. (13246)
Do you wish to continue waiting..... Type y to continue (y/n). (13247)
y
Usr     Name     Type      Pid Timestamp
        5  docusr1 Usr         10336 1129580580

Connections with timestamps older than 1129752317  exist. (13246)
Do you wish to continue waiting..... Type y to continue (y/n). (13247)
y
Index Activate: Index inact3  No. 33  - 0 keys added. (13244)
Index Activate: Index inact3  No. 33  - 0 keys processed. (13245)
Index Activate: Index inact3 has been activated.
$
```

Once the IDXACTIVATE command completes, the index is active and all users can access it.

Database Client Notification

Database Client Notification is a communication mechanism between clients and the database, informing clients of the need to refresh their schema cache.

Prior to OpenEdge Release 11.7, certain database utilities require clients to disconnect, even though the utility can execute while the database is online. The disconnect of the clients is required because these utilities update index schema. Database Client Notification allows clients to remain connected and refresh their schema in step with the utility execution. Notification enables the index activation processes to move to completion more rapidly.

Utilities supported by database client notification are:

- PROUTIL IDXACTIVATE
- PROUTIL LOAD (with `buildindexes`)
- PROUTIL MTIDXBUILD
- PROUTIL TPIDXBUILD

Database client notification is controlled by a server startup parameter that indicates how often a client polls for notification. The parameter is specified as follows:

```
-usernotifytime polltime
```

The *polltime* value indicates, in seconds, how often a client polls for notification of a schema change. The default is 0, indicating that no polling occurs. The minimum non-zero value is 1; the maximum is 86400 (for 24 hours).

The *polltime* value can be changed on a running server through PROMON or the _DbParams VST. See Changing Database Client Notification poll time on page 322, for more information.

If *polltime* is zero (0), the behavior of the supported utilities is the same as previous releases. The utility will not proceed until clients with an earlier schema timestamp are disconnected.

At connect time, every client establishes a Database Client Connection thread to poll for notifications. The client always polls one time. If the value of *polltime* is zero (0), the client never polls again. If the value of *polltime* is non-zero, the client polls for notification after *polltime* seconds has elapsed. If the client has a pending notification when it polls, the client re-caches its index schema as soon as processing allows (for example, at the end of a transaction), and clears the notification, letting the utility know it can proceed with its processing. Depending on the activity of the client, there can be a delay longer than *polltime* seconds before the client can respond to the notification.

Changing Database Client Notification poll time

Database Client Notification is a communication mechanism between clients and the database, informing clients of the need to refresh their schema cache. Notification enables timelier completion of certain database administrative actions, such as activating an inactive index. The timelier completion makes the result of the administrative change, such as an active index, available more quickly for application use.

The Database Client Notification startup parameter indicates, in seconds, how often a client polls for notification of a schema change. The default is 0, indicating that no polling occurs. The minimum non-zero value is 1; the maximum is 86400 (for 24 hours).

You can change the value of the startup parameter at runtime with one of the following methods:

PROMON

To change the Database Client Notification poll time with PROMON, make the following selections:

1. From the PROMON main menu enter, **R&D**. The R&D main menu appears.
2. From the R&D main menu, choose **Administrative Funtions** by entering **4**.
3. From the **Administrative Funtions** menu, choose **Server Options** by entering **7**.
4. From the **Server Options** menu, choose **User notify time** by entering **8**. You are prompted to enter a new poll time as follows:

```
Enter Attention poll time (1 to 86400):
```

5. Enter the new value, and return.

_DbParams VST

To change the Database Client Notification poll time with the _DbParams VST, use the following ABL example as a guide:

1. Query the current value:

```
FIND FIRST _DbParams WHERE _DbParams._DbParams-Name = "-usernotifytime".
DISPLAY "User notification poll time: " _DbParams._DbParams-Value.
```

2. Update the current value:

```
DO TRANSACTION:
  FIND FIRST _DbParams WHERE _DbParams._DbParams-Name = "-usernotifytime".
  ASSIGN _DbParams._DbParams-value = "3600". /* one hour */
END.
```

Virtual system tables

Virtual system tables (VSTs) provide ABL and SQL access to system statistics. You can use this information to help analyze database and application design, as well as to determine what factors are impacting database performance. The virtual system tables, or schema tables, have no physical records until the database manager generates them at run time. For a detailed description of each table, see Virtual System Tables on page 863.

15

Maintaining Database Structure

Once you create and start a database, you must manage the database so that it operates efficiently and meets the needs of users. This chapter describes methods to manage the database structure and alter it as necessary to improve storage and performance.

For details, see the following topics:

- OpenEdge Structure utility
- OpenEdge Structure Statistics utility
- OpenEdge Structure List utility
- Using the OpenEdge Structure Add utility
- OpenEdge Structure Add Online utility
- Area numbers
- Validating structure description files
- Using the OpenEdge Structure Remove utility
- Maintaining indexes and tables
- Performing chain analysis online
- Using virtual system tables

OpenEdge Structure utility

Using the OpenEdge Structure (PROSTRCT) utility, you can scale your database to meet business requirements. For example, if your business requires high availability, use PROSTRCT to reorganize tables and indexes while users operate against the database. The following sections detail how to maintain the structure of your OpenEdge database, including when to use PROSTRCT and PROUTIL.

OpenEdge Structure Statistics utility

Use the Structure Statistics utility (PROSTRCT STATISTICS) to monitor database growth and the availability of free blocks for storing additional data. For example:

```
prostrct statistics db-name
```

You specify the database whose storage information you want and the PROSTRCT STATISTICS utility displays information about:

- The database name
- The primary database block size and the before-image and after-image block sizes
- The storage area and each extent within it
- The total number of active blocks allocated for each data area
- The total number of empty blocks in each data area
- The total number of extent blocks for each data area
- The total number of blocks for each data area
- The total records per block for each data area
- The total number of all blocks (active, data, free, empty, extent, and total blocks)
- The date and time of the last full database backup

Running PROSTRCT STATISTICS against an online database gives you a snapshot of the database state at that moment in time.

For example, this command displays the PROSTRCT STATISTICS command and part of the database storage information output for the database `/usr1/doc/sample.db`:

```
$ prostrct statistics sample

WARNING: Before-image file of database sample is not truncated. (1552)
The last backup was not completed successfully. (1553)

Storage Utilization Statistics

Database: sample

Primary data block size: 1024
          BI block size: 8192
          AI block size: 8192
```

```
Database Phyical structure information

Statistics for Area: Control Area

  Files in Area: Control Area
     /usr1/doc/sample.db  32768

 Database Block Usage for Area: Control Area

Active blocks: 6
   Empty blocks: 26
  Extent blocks: 1
   Total blocks: 32
  Records/Block: 32

  Statistics for Area: Primary Recovery Area

  Files in Area: Primary Recovery Area
     /usr2/doc_bi/sample.b1  31064064

  Statistics for Area: Schema Area

  Files in Area: Schema Area
     /usr1/doc/sample.d1  950272

Database Block Usage for Area: Schema Area

  Active blocks: 884
   Empty blocks: 44
  Extent blocks: 1
   Total blocks: 928
  Records/Block: 32

Statistics for Area: Employee

  Files in Area: Employee
     /usr1/doc/sample_7.d1  327680
     /usr1/doc/sample_7.d2  32768

  Database Block Usage for Area: Employee

  Active blocks: 28
   Empty blocks: 324
  Extent blocks: 2
   Total blocks: 352
  Records/Block: 32
     .
     .
     .
Statistics for Area: After Image Area 1

  Files in Area: After Image Area 1
     /usr3/doc_ai/sample.a1  524288

  Statistics for Area: After Image Area 2

  Files in Area: After Image Area 2
     /usr3/doc_ai/sample.a2  524288

  Statistics for Area: After Image Area 3

  Files in Area: After Image Area 3
     /usr3/doc_ai/sample.a3  93454336

Statistics for Area: Audit Area

  Files in Area: Audit Area
     /usr1/doc/sample_20.d1  3440640
```

```
Database Block Usage for Area: Audit Area

  Active blocks: 3311
   Empty blocks: 49
  Extent blocks: 1
   Total blocks: 3360
  Records/Block: 32

  Statistics for Area: Audit Index

  Files in Area: Audit Index
     /usr1/doc/sample_22.d1  1540096

Database Block Usage for Area: Audit Index

  Active blocks: 1459
   Empty blocks: 45
  Extent blocks: 1
   Total blocks: 1504
  Records/Block: 32

 Database Block Usage Summary

  Active blocks: 8320
            Data: 8141
            Free: 179
   Empty blocks: 1184
  Extent blocks: 16
   Total blocks: 9504

Last full database backup on Tue Sep 13 15:28:07 2005. (6942)
```

If a full backup has not yet been performed against `service.db`, the message would read "NO FULL BACKUP HAS BEEN DONE."

OpenEdge Structure List utility

Your structure description file must reflect the current information in your database control area to ensure continuity and ease database management. Therefore, update the structure description file any time you make changes to the structure of a database, such as adding, moving, or removing extents. Use the Structure List (PROSTRCT LIST) utility to update your structure description file. PROSTRCT LIST produces a new structure description file for the database you name, using the current information stored in the database control area.

Note: You can use PROSTRCT LIST with an online database.

To update the structure description file with the current information stored in the database control area, use the PROSTRCT LIST utility:

```
prostrct list db-name [ structure-description-file ]
```

In the command, *db-name* specifies the database whose structure description file you want to update, and *structure-description-file* specifies the structure description file to create. If you do not specify the structure description file, PROSTRCT LIST uses the base name of the database and appends a `.st` extension. It replaces an existing file of the same name.

For example, to update the structure description file for `/user/joe/service`, enter the following command:

```
prostrct list /user/joe/service
```

PROSTRCT LIST displays the information it produces in the structure description file, including storage area type, storage area name, records per block, extent pathname, and extent type, either fixed or variable (a size of 0 indicates a variable-length extent). For example:

```
AreaName: ControlArea, Type 6, Block Size 4096, Extents 1, Records/Blck 32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 32 KByte, Name: /user/joe/service.db

AreaName: Primary Recovery Area, Type 3, Block Size 8192, Extents 1
   Ext # 1, Type VARIABLE, Size 2176 KByte, Name: /usrer/joe/service.b1

AreaName: Transaction Log Area, Type 4, Block Size 16384, Extents 1
   Ext # 1, Type FIXED   , Size 1024, Name:/user/joe/service.t1

AreaName: Schema Area, Type 6, Block Size 1024, Extents 1, Records/Blck 32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 960 KByte, Name: /user/joe/service.d1

AreaName: After Image Area 1, Type 7, Block Size 8192, Extents 1
   Ext # 1, Type FIXED   , Size 32, Name: /user/joe/service.a1

AreaName: service, Type 6, Blck Size 1024, Extents 6, Records/Block 32,
Cluster Size 1
   Ext # 1, Type FIXED   , Size 32, Name:/user/joe/service_8.d1
   Ext # 2, Type FIXED   , Size 32, Name:/user/joe/service_8.d2
   Ext # 3, Type FIXED   , Size 32, Name:/user/joe/service_8.d3
   Ext # 4, Type FIXED   , Size 32, Name:/user/joe/service_8.d4
   Ext # 5, Type FIXED   , Size 32, Name:/user/joe/service_8.d5
   Ext # 6, Type VARIABLE , Size 0, Name:/user/joe/service_8.d6

AreaName: joe, Type 6, Block Size 1024, Extents 1, Records/Block 32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 0, Name: /user/joe/yy/z/service_1000.d1
```

Using the OpenEdge Structure Add utility

Use the Structure Add (PROSTRCT ADD) utility to add storage areas and extents to an existing database. The database must be offline before you can use PROSTRCT ADD, however PROSTRCT ADDONLINE is available for use with an online database. For information on PROSTRCT ADDONLINE, see OpenEdge Structure Add Online utility on page 332.

To add storage areas to an existing database using PROSTRCT ADD:

1. Back up your database.
2. Create a new structure description file that contains only information about the areas you want to add.

   The following is an example structure description file:

```
# add.st
#
d "chris",128 . f 1024
d "chris" .
```

Note: To avoid overwriting the `.st` file for your existing database, the name of this `.st` file must be different from the existing `.st` file for the database. For example, name the new structure description file `add.st`.

3. Use the PROSTRCT Utility with the ADD qualifier, specifying the `.st` file you created in Step 2.

   The following is a sample command:

```
prostrct add service add.st
```

   PROSTRCT ADD adds the new extents or storage areas and extents to the existing database control area, and outputs descriptive information, such as:

```
Formatting extents:
   size              area name           path name
   1024                chris        /user/joe/service_9.d1  00:00:00
     32                chris        /user/joe/service_9.d2  00:00:01
```

4. Generate an update structure definition file for your database with PROSTRCT LIST.

   The following is a sample command:

```
prostrct list service service.st
```

After you modify the structure of your database (adding or removing extents), run PROSTRCT LIST. PROSTRCT LIST automatically creates a structure description file for your database if one does not exist, or overwrites an existing `.st` file of the same name to reflect the changes you just made to the structure of your database.

PROSTRCT ADD and pathnames

When you use PROSTRCT ADD to add additional storage areas or extents to a relative path database, the relative path database is converted to an absolute path database. For example, if you add an additional application data storage area named `Misc` to the relative path database `example1`, PROSTRCT ADD converts the relative path database to an absolute path database. Use PROSTRCT LIST to update the `.st` file with the added extents. For example:

```
prostrct add example1 add.st
prostrct list example1
```

In the following sample output of the PROSTRCT LIST utility, note the absolute pathname (`/usr1/example1.db`) of the modified database:

```
Area Name: Control Area, Type6, BlockSize 4096, Extents 1, Records/Block32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 0, Name: /usr1/example1.db
```

```
Area Name: Primary Recovery Area, Type3, BlockSize 8192, Extents 1
   Ext # 1, Type VARIABLE, Size 0, Name: /usr1/example1.b1

Area Name: Schema Area, Type6, BlockSize 4096, Extents 1, Records/Block 32,
Cluster Size 1
   Ext # 1, Type VARIABLE, Size 0, Name: /usr1/example1.d1

Area Name: Employee, Type6, BlockSize 1024, Extents 2, Records/Block 32,
Cluster Size 8
   Ext # 1, Type FIXED   , Size 320, Name: /usr1/example1_7.d1
   Ext # 2, Type VARIABLE, Size 0, Name: /usr1/example1_7.d2

Area Name: Inventory, Type6, BlockSize 1024, Extents 2, Records/Block 32,
Cluster Size 64
   Ext # 1, Type FIXED   , Size 608, Name: /usr/example1_8.d1
   Ext # 2, Type VARIABLE, Size 0, Name: /usr1/example1_8.d2

Area Name: Cust_Data, Type6, BlockSize 1024, Extents 2, Records/Block 32,
Cluster Size 64
   Ext # 1, Type FIXED   , Size 320, Name: /usr1/example1_9.d1
   Ext # 2, Type VARIABLE, Size 0, Name: /usr1/example1_9.d2

Area Name: Cust_Index, Type 6, BlockSize 1024, Extents 2, Records/Block 32
   Ext # 1, Type FIXED   , Size 320, Name: /usr1/example1_10.d1
   Ext # 2, Type VARIABLE, Size 0, Name: /usr1/example1_10.d2

Area Name: Order, Type6, BlockSize 1024, Extents 2, Records/Block 32,
Cluster Size 64
   Ext # 1, Type FIXED   , Size 1280, Name: /usr1/example1_11.d1
   Ext # 2, Type VARIABLE, Size 0, Name: /usr1/example1_11.d2

Area Name: Misc, Type6, BlockSize 1024, Extents 2, Records/Block 32
   Ext # 1, Type FIXED   , Size 320, Name: /usr1/example1_12.d1
   Ext # 2, Type VARIABLE, Size 0, Name: /usr1/example1_12.d2
```

Adding extents to existing storage areas

To add extents to an existing area, use the PROSTRCT utility with the ADD qualifier.

Note: You can only add transaction-log (TL) extents when the database is offline.

To add extents to an existing database using PROSTRCT ADD:

1. Back up your database.
2. Create a new structure description file that contains only information about the areas you want to add.

   For example:

```
# add.st
#
d "chris",128 .
```

   Note: To avoid overwriting the `.st` file for your existing database, the name of this `.st` file must be different from the existing `.st` file for the database. For example, name the new structure description file `add.st`.

3. Use the PROSTRCT utility with the ADD qualifier, specifying the `.st` file you created in Step 2.

For example:

```
prostrct add service add.st
```

PROSTRCT ADD adds the new extents or storage areas and extents to the existing database control area and outputs descriptive information, such as:

```
Formatting extents:
   size            area name         path name
     0               chris        /user/joe/service_9.d3  00:00:01
```

PROSTRCT ADD converts your previous variable-length extent to a fixed-length extent at its current size.

4. Generate an update structure definition file for your database PROSTRCT LIST.

   For example:

```
prostrct list service service.st
```

After you modify the structure of your database (adding or removing extents), run PROSTRCT LIST. PROSTRCT LIST automatically creates a structure description file for your database if one does not exist, or overwrites an existing `.st` file of the same name to reflect the changes you just made to the structure of your database.

OpenEdge Structure Add Online utility

Use the Structure Add Online (PROSTRCT ADDONLINE) utility to add storage areas and extents to an existing database while it is online. PROSTRCT ADDONLINE allows a database administrator to add data areas and extents, before-image areas and extents, and after-image areas and extents without incurring downtime.

PROSTRCT ADDONLINE has the following restrictions:

- You can not have more than one instance of PROSTRCT ADDONLINE executing at a single time.
- All connected users must have sufficient privileges to access the newly created extents. If currently connected users will not have sufficient privileges to access the new extents, you may begin the ADDONLINE, but the users must be disconnected before the ADDONLINE can complete.

PROSTRCT ADDONLINE follows this general procedure:

1. Check status of all connected users. If any connected users do not have sufficient privileges to access the new extents, you are informed of the users and the risks with proceeding. You are prompted to continue, as shown:

```
Usr Name Type Pid
01  usr1 Usr  1234
102 usr2 Usr  5678
...
There exist connections to the database which will have
problems opening newly added extents.
These connections may abort adding extent online later.
Do you want to continue adding extents online? (y/n)
```

Answer **Y** to continue the ADDONLINE. Answer **N** to abort.

2. Create the physical files associated with the extents. This is the longest step, but requires no locks so connected users are not impacted.
3. Re-check the status of all connected users. You are prompted as follows:

```
Usr Name Type Pid
01  usr1 Usr  1234
102 usr2 Usr  5678
...
There exist connections to the database which will have
problems opening newly added extents.
Do you wish to recheck user permissions and continue adding extents online?
```

   Answer **Y** to proceed to recheck. Answer **N** to abort ADDONLINE.

   If you answer **Y**, ADDONLINE will loop, performing the privilege check, and prompting you to continue as long as there are under-privileged users connected.

   If any under-privileged users are connected, ADDONLINE cannot proceed. You must disconnect the users with PROSHUT, wait for them to disconnect, or cancel the ADDONLINE process.

4. Acquire the area admin lock.
5. If adding an extent to an existing data area, convert the current variable-length extent to a fixed length extent.
6. Add the new extent and size information to the area in the database control area.
7. Release the area admin lock.

If ADDONLINE fails, the newly created files are deleted.

Use the following steps to add storage areas to an existing database using PROSTRCT ADDONLINE:

1. Create a new structure description file that contains only information about the areas you want to add.

   For example:

```
# FILE: add.st
## Add additional before-image extents
b  .    f 1024
b  .
## Add additional extents to the "Employee" area
d "Employee":7,32;1  . f 4096
d "Employee":7,32;1  .
## Add after-image areas
a  . f 512
a  . f 512
a  .
```

   Note: To avoid overwriting the `.st` file for your existing database, the name of this `.st` file must be different from the existing `.st` file for the database. For example, name the new structure description file `add.st`.

2. Validate your structure definition file.

For example:

```
prostrct addonline service add.st -validate
```

Adding the `-validate` parameter directs PROSTRCT to check the syntax and contents of your `.st` file, without performing the add. For complete information on the use of the `-validate` parameter, see Validating structure description files on page 341.

3. Add the new areas and extents. Use the PROSTRCT Utility with the ADDONLINE qualifier, specifying the `.st` file you created in Step 1 on page 333.

   For example:

```
prostrct addonline service add.st
```

   PROSTRCT ADDONLINE adds the new extents or storage areas and extents to the existing database control area, and outputs descriptive information, such as:

```
Formatting extents:
   size                area name   path name
    128    Primary Recovery Area /user/joe/service.b3 00:00:02
     16    Primary Recovery Area /user/joe/service.b4 00:00:02
   4096                 Employee /user/joe/service_7.d5 00:00:02
     32                 Employee /user/joe/service_7.d6 00:00:04
     64      After Image Area 10 /user/joe/service.a10 00:00:00
     64      After Image Area 11 /user/joe/service.a11 00:00:00
     16      After Image Area 12 /user/joe/service.a12 00:00:00
Enabling extents:
   size                area name   path name (13170)
    128    Primary Recovery Area /user/joe/service.b3 00:00:00
     16    Primary Recovery Area /user/joe/service.b4 00:00:00
   4096                 Employee /user/joe/service_7.d5 00:00:00
     32                 Employee /user/joe/service_7.d6 00:00:00
     64      After Image Area 10 /user/joe/service.a10 00:00:01
     64      After Image Area 11 /user/joe/service.a11 00:00:00
     16      After Image Area 12 /user/joe/service.a12 00:00:00
```

4. Generate an update structure definition file for your database PROSTRCT LIST.

   For example:

```
prostrct list service service.st
```

After you modify the structure of your database (adding or removing extents), run PROSTRCT LIST. PROSTRCT LIST automatically creates a structure description file for your database if one does not exist, or overwrites an existing `.st` file of the same name to reflect the changes you just made to the structure of your database.

Area numbers

An OpenEdge database can have area numbers up to 32,000. Assigning area numbers is optional and the maximum can be controlled with a startup parameter. Area numbers are discussed in the following sections:

- Area number assignment on page 335
- Trimming unused area memory on page 338
- Error checking on page 340

Area number assignment

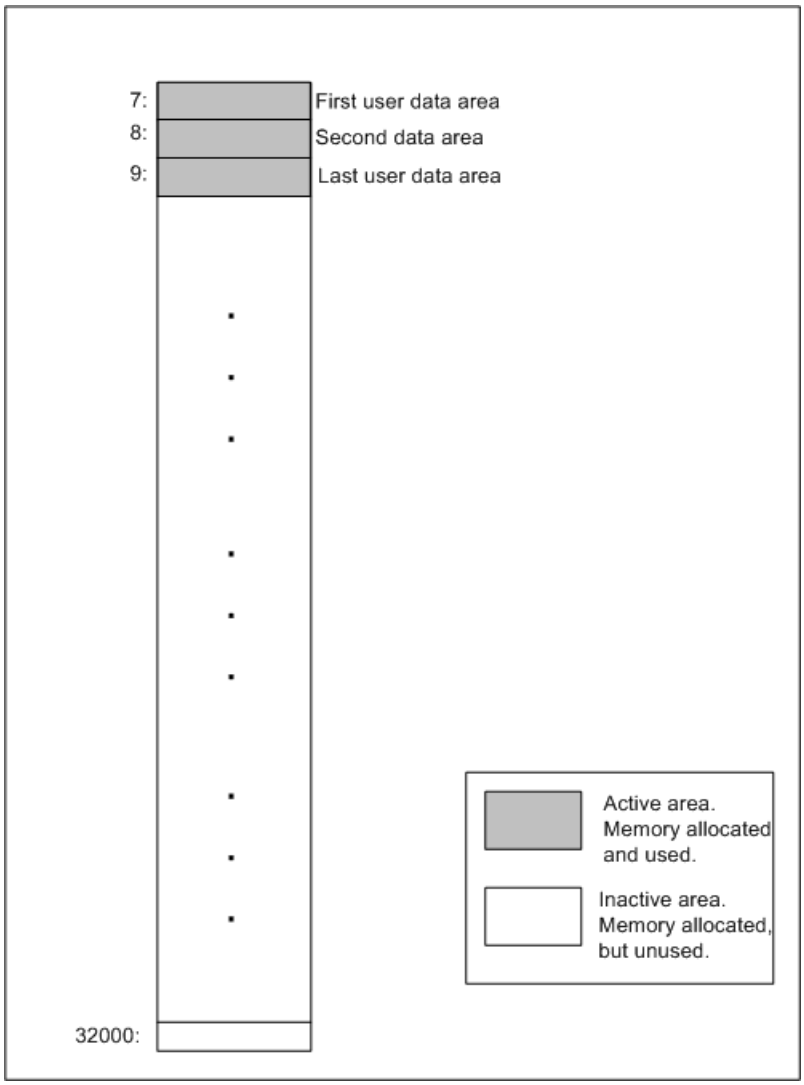
If you create your database areas without assigning area numbers, OpenEdge assigns the areas sequentially starting at area number 7. The following example shows a structure definition file without area numbers:

```
#
b . f 1024
b .
#
d "Schema Area":6,32;1 .
#
#
d "First user data area",32;64 . f 4096
d "First user data area",32;64 .
#
d "Second user data area",32;64 . f 4096
d "Second user data area",32;64 .
#
d "Last user data area",32;64 . f 4096
d "Last user data area",32;64 .
#
```

A database created with PROSTRCT using this structure file as input, creates the areas sequentially.

The following figure shows the resulting area array where the areas are numbered sequentially.

Figure 42: Sequential area allocation



Assigning specific area numbers to the user data areas in your structure file can improve readability, but will prevent the ability to trim the maximum amount of unused memory in your area array.

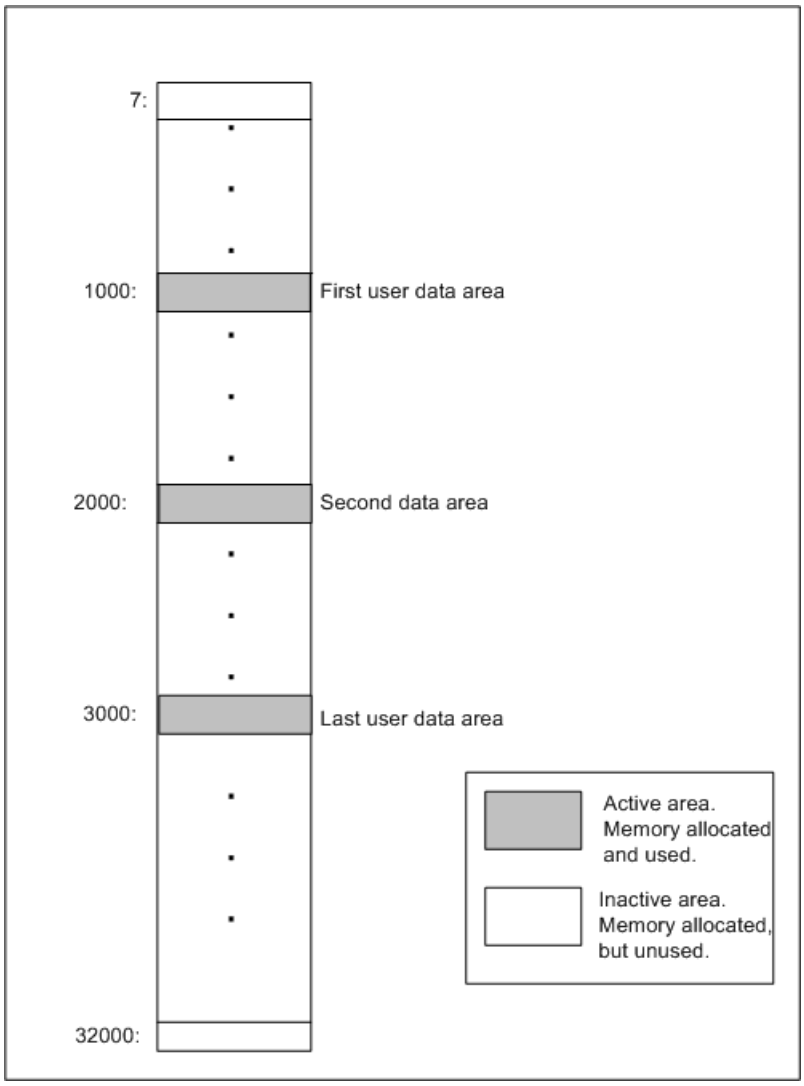
The following example shows a structure file with three data areas assigned area numbers 1000, 2000, and 3000:

```
#
b . f 1024
b .
#
d "Schema Area":6,32;1 .
#
#
d "First user data area":1000,32;64 . f 4096
d "First user data area":1000,32;64 .
#
d "Second user data area":2000,32;64 . f 4096
d "Second user data area":2000,32;64 .
#
d "Last user data area":3000,32;64 . f 4096
d "Last user data area":3000,32;64 .
#
```

A database created with PROSTRCT using this structure file as input, creates the areas as they are defined.

The following figure shows the allocation of the area array for the database with areas numbered 1000, 2000, and 3000.

Figure 43: Area allocation with assigned area numbers



If minimizing memory consumption is a priority, you can trim the unused memory for the areas numbered 3001 to 32,000 with the `-maxAreas` startup parameter. However, the memory for the unused areas in between the allocated areas cannot be trimmed. See Trimming unused area memory on page 338 for more information.

Trimming unused area memory

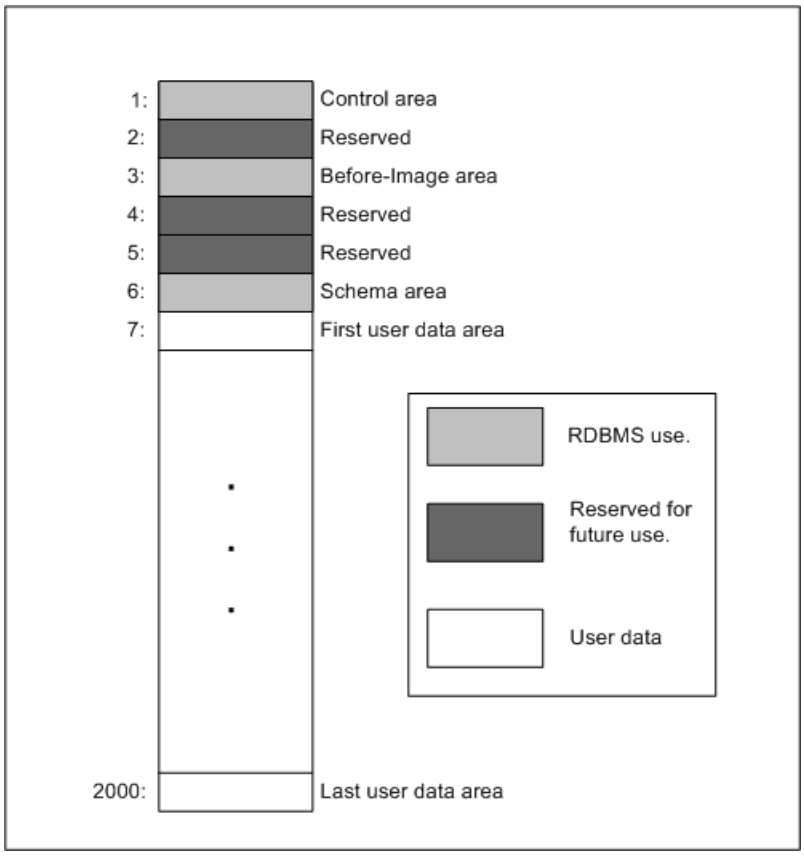
If you know that you will never need to add areas beyond your current maximum area number while the database is online, you can trim the area array with the `-maxAreas` server startup parameter. Trimming the area array saves memory within the server process. For a complete description of the `-maxAreas` syntax, see Maximum area number (-maxAreas) on page 510.

If you have a database with the following structure description file, trimming the area array with `-maxAreas 2000` results in the allocation of a smaller area array:

```
#
b . f 1024
b .
#
d "Schema Area":6,32;64 .
#
#
d "First user data area":7,32;64 . f 4096
d "First user data area":7,32;64 .
#
.
.
.
#
d "Last user data area":2000,32;64 . f 4096
d "Last user data area":2000,32;64 .
#
```

The following figure shows the layout of areas array in memory for a database created from this structure file, and trimmed by including `-maxAreas 2000` on the server startup.

Figure 44: Area array trimmed by -maxAreas startup parameter



Setting `-maxAreas` equal to your largest area number leaves no room for potential growth beyond that area number. You can save unused memory, and provide room for growth, by setting `-maxAreas` slightly higher than your maximum area number.

If during the time your database is online, you need to add areas beyond the current maximum area number specified by `-maxAreas`, you must shutdown and restart your database. Your options are:

- Shutdown your database and restart with a larger `-maxAreas` setting.
- Shutdown your database and remove the `-maxAreas` parameter from the startup command. This will allow the maximum area number, 32,000.

Error checking

Agreement between the defined area numbers of the database and the `-maxAreas` startup parameter is verified at two points:

- Server startup on page 340
- Adding areas online on page 340

Server startup

If the `-maxAreas` parameter is included at server startup, the server verifies that the maximum area number defined for the database is less than or equal to the parameter value. If `-maxAreas` *n* is smaller than the maximum defined area number, the server will not start and will issue an error message. The following example shows the error message created when `-maxAreas 200` was specified on the server startup of a database with a maximum area number of 2000:

```
4:21:45 BROKER  0: Startup parameter -maxAreas 200 is too small to open a
database with area number 2000 (13795)
```

Adding areas online

If the server has been successfully started with the `-maxAreas` parameter, PROSTRCT ADDONLINE will verify that the area numbers of the areas being added do not exceed the specified maximum area number. The following output shows the error that results from attempting to add areas with numbers higher than the `-maxAreas` parameter:

```
$ proserve docsample -maxAreas 2000
$
$ cat beyondmax.st
#
# Attempt to add an area beyond -maxarea
#
d "MaxAreaTest must fail":2999,32;64 .  f 4096
d "MaxAreaTest must fail":2999,32;64 .

$ prostrct addonline docsample beyondmax.st

Syntax error in structure file:

d "MaxAreaTest must fail":2999,32;64 .  f 4096

Invalid area number specified. (6826)
Error occurred on line 4. (6819)
prostrct add FAILED. (12867)
$
```

Validating structure description files

Prior to creating or updating a database with PROSTRCT, you can verify that your structure definition file (`.st`) is correct by adding the `-validate` parameter to your PROSTRCT CREATE, PROSTRCT ADD, or PROSTRCT ADDONLINE commands. When `-validate` is added to your PROSTRCT command, the structure definition file is examined for accurate syntax and valid content for the specified operation, and PROSTRCT determines if there is currently sufficient disk space to create the new extents. Any errors encountered in the structure definition file are reported. The add or create will not execute, even if there are no errors in the structure definition file.

Validation of a structure definition syntax includes the following checks:

- Verify that every line in the file is either blank, a comment, or a proper extent definition.
- Verify that every non-blank line begins with a comment character (`*`, `:`, `#`) or an extent type indicator (`a`, `b`, `d`, `t`).
- Verify that if the extent definition includes area information, it begins with an area name contained in double quotes (`" "`).
- Verify that if an area number, records per block, or blocks per cluster field is defined for one extent, they are the same for all extents defined for that area.
- Verify that if delimiters are included, they adhere to the following rules:
  - A colon (`:`) is followed by the area number
  - A comma (`,`) is followed by the records per block value
  - A semicolon (`;`) is followed by the blocks per cluster value
- Verify that if the cluster size is specified, it is one of the following values: 0, 1, 8, 64, 512.
- Verify that a path specification is included for each extent, and that any path with a space in the name is enclosed in double quotes(`" "`), and preceded by an exclamation point (`!`).
- Verify that size information for each extent is either blank or includes a type (`f` or `v`) and a size in 1K units.
- Verify that size information of an extent, when specified, is a multiple of 16 times the block size and larger than 32; for Type II areas, also verify the size is large enough to contain at least one cluster. These sizes are automatically rounded up when the files are created, but validation issues a warning that the rounding will occur.

Validation also verifies that for each extent defined, there is sufficient disk space available to create the file.

For information on the syntax of the structure definition file, see Creating a structure description file on page 43.

Validation also checks that the following utility restrictions are adhered to:

- For all utilities, the directories where extents are specified must exist
- For PROSTRCT CREATE, the database cannot already exist
- For PROSTRCT ADD and ADDONLINE, you cannot add transaction log extents when two-phase commit is enabled

Using the OpenEdge Structure Remove utility

Use the Structure Remove (PROSTRCT REMOVE) utility to remove extents from storage areas. After all the extents are removed from a storage area, PROSTRCT REMOVE removes the storage area.

Us the following steps to remove storage areas:

1. Truncate the area, based on its type:

   - If the extent to be removed is in the BI area, use the PROUTIL TRUNCATE BI utility to truncate the primary recovery area to be removed. For example:

     ```
     proutil db-name -C truncate bi
     ```

   - If the storage area to be removed is an application data area, use the PROUTIL TRUNCATE AREA utility to truncate the application data area. For example:

     ```
     proutil db-name -C truncate area area-name
     ```

   Note: You must disable after-imaging before you can remove an AI extent. You must also disable two-phase commit before you can remove a transaction-log (TL) extent.

   For more information about truncating areas, see PROUTIL TRUNCATE AREA qualifier on page 775.

2. Use PROSTRCT REMOVE to remove extents from the storage area.

   For example:

   ```
   prostrct remove db-nameextent-type-token [area-name]
   ```

   Note: Use the *area-name* parameter only to remove application data extents. If the area name contains a space, supply double-quotes around the area name. For example: `"test data."`

   You can remove one extent at a time. After you have removed all of the extents from a storage area, PROSTRCT REMOVE removes the storage area and outputs descriptive information such as:

   ```
   solaris:100a$ prostrct remove service d chris
   /user/joe/service_9.d3 successfully removed
   solaris:100a$ prostrct remove service d chris
   /user/joe/service_9.d2 successfully removed
   solaris:100a$ prostrct remove service d chris
   /user/joe/service_9.d1 successfully removed
   ```

3. Run PROSTRCT LIST after removing any areas. PROSTRCT LIST will overwrite your existing `.st` file to reflect the changes made to the structure of your database.

Maintaining indexes and tables

You can add tables and indexes to existing storage areas in a database using the **Create Table** and **Create Index** commands from the Data Dictionary. However, you cannot add tables or indexes to existing storage areas using the **Modify Table** command or the **Index property** sheet. The following sections detail how to move tables and indexes and how to compact indexes.

Moving tables

Use the PROUTIL TABLEMOVE utility to move a table and its associated indexes from one storage area to another while the database remains online. For example:

```
proutil db-name -C tablemove [owner-name.] table-nametable-area [index-area]
```

Note: For the complete syntax description, see PROUTIL TABLEMOVE qualifier on page 770. The `_UserStatus` virtual system table (VST) displays the utility's progress. For more information on VSTs, see Virtual System Tables on page 863.

If you omit the *index-area* parameter, the indexes associated with the table will **not** be moved.

Moving the records of a table from one area to another invalidates all the ROWIDs and indexes of the table. Therefore, the indexes are rebuilt automatically by the utility whether you move them or not. You can move the indexes to an application data area other than the one to which you are moving the table. If you want to move only the indexes of a table to a separate application data area, use the PROUTIL IDXMOVE utility.

Moving a table's indexes with the TABLEMOVE qualifier is more efficient than moving a table separately and then moving the indexes with the IDXMOVE utility. Moving a table separately from its indexes wastes more disk space and causes the indexes to be rebuilt twice, which also takes longer.

Note: While you can move tables online, no access to the table or its indexes is recommended during the move. The utility acquires an EXCLUSIVE lock on the table while it is in the process of moving. An application that reads the table with an explicit NO-LOCK might be able to read records, but in some cases can get the wrong results, since the table move operation makes many changes to the indexes. Run the utility during a period when the system is relatively idle, or when users are doing work that does not access the table.

The PROUTIL TABLEMOVE utility operates in phases:

- **Phase 1** — The records are moved to the new area and a new primary key is built.
- **Phase 2** — All the secondary indexes are built:
  - If you did not specify the *index-area* parameter, then the indexes are rebuilt in their original area.
  - If you did specify the *index-area* parameter, then all the indexes are moved to the new area where they are rebuilt.
- **Phase 3** — All the records in the old area are removed.
- **Phase 4** — All the old indexes are removed and the _StorageObject records of the indexes and the table are updated.

Note: Although PROUTIL TABLEMOVE operates in phases, it moves a table and its indexes in a single transaction. To allow a full recovery to occur when a transaction is interrupted, every move and delete of each individual record is logged. As a result, moving a table requires the BI Area to be several times larger than the combined size of the table and its indexes. Therefore, before moving your table, determine if your available disk capacity is sufficient to support a variable BI extent that might grow to more than three times the size of the table and its indexes.

Moving indexes

Use the PROUTIL IDXMOVE utility to move an index from one application data area to another while the database remains online. You might be able to improve performance by moving indexes that are heavily used to an application data area on a faster disk. For example:

```
proutil db-name -C idxmove [owner-name.] indexname area-name
```

Note: For the complete syntax description, see PROUTIL IDXMOVE qualifier on page 734.

The PROUTIL IDXMOVE utility operates in two phases:

- **Phase 1** — The new index is being constructed in the new area. The old index remains in the old area, and all users can continue to use the index for read operations.
- **Phase 2** — The old index is being removed, and all the blocks of the old index are being moved to the free block chain. For a large index this phase can take a significant amount of time. During this phase, all operations on the index are blocked until the new index is available; users accessing the index might experience a freeze in their application.

The `_UserStatus` virtual system table displays the utility's progress.

Note: While you can move indexes online, no writes to the table or its indexes are allowed during the move. The IDXMOVE utility acquires a SHARE lock on the table, which blocks all attempts to modify records in the table. Run the utility during a period when the system is relatively idle, or when users are doing work that does not access the table.

Compacting indexes

When the DBANALYS utility indicates that space utilization of an index is reduced to 60 percent or less, use the PROUTIL IDXCOMPACT utility to perform index compaction online. Performing index compaction increases space utilization of the index block to the compacting percentage specified. For example:

```
proutil db-name -C idxcompact [owner-name.] table-name.index-name [n ]
```

Note: For the complete syntax description, see PROUTIL IDXCOMPACT qualifier on page 726

Performing index compaction reduces the number of blocks in the B-tree and possibly the number of B-tree levels, which improves query performance.

The index compacting utility operates in phases:

- **Phase 1** — If the index is a unique index, the delete chain is scanned and the index blocks are cleaned up by removing deleted entries.
- **Phase 2** — The nonleaf levels of the B-tree are compacted starting at the root working toward the leaf level.
- **Phase 3** — The leaf level is compacted.

The `_UserStatus` virtual system table displays the utility's progress.

Note: Because index compacting is performed online, other users can use the index simultaneously for read or write operation with no restrictions. Index compacting only locks one to three index blocks at a time, for a short time. This allows full concurrency.

Performing chain analysis online

OpenEdge can perform chain analysis **online** in one of two ways. You can use the `dbanalys` qualifier or the `chanalys` qualifier, as shown in the following examples:

- DBANALYS

```
proutil db-name -C dbanalys
```

  Performing chain analysis using `dbanalys` accomplishes the following:

  - Executes a minimal level online chain analysis
  - Runs as part of database analysis, reporting on the lengths, but not the content of the cluster free, free, RM, and index delete chains
  - Produces minimal performance impact
  - Does not perform sanity checks of offline chain analysis

- CHANALYS

```
proutil db-name -C chanalys
```

  Performing chain analysis online using `chanalys` accomplishes the following:

  - Executes an intermediate level online chain analysis
  - Produces minimal performance impact due to minimal locking
  - Reports `dbkeys` of each chain entry
  - Online chain analysis may exit prematurely if:
    - A user presses **CTRL+C**
    - A block is encountered twice in the chain
    - The number of blocks processed is greater than the number reported in the object block
    - The block is no longer on the chain

Important considerations when performing chain analysis online

When using the CHANALYS qualifier, `dbkeys` in the chain are displayed in addition to the lengths of the chains. When this is used in an online database, the following inconsistencies might appear:

- Duplicate `dbkeys` in the chain
- Fewer blocks in the chain than reported in the chain length
- The actual chain may contain more `dbkeys` than are reported

These inconsistencies do not automatically indicate a problem with the chain; it may be the result of reporting on a changing chain. For more information, see PROUTIL Utility on page 629.

Using virtual system tables

Depending on the size and complexity of your database, some online database administration utilities can take a significant amount of time to complete. You can use the _UserStatus virtual system table (VST) to monitor the state of the following administration utilities that can be performed online:

- **IDXANALYS** — Displays information about index blocks
- **IDXCOMPACT** — Increases space utilization of an index block
- **IDXFIX** — Detects corrupt indexes and records with a missing or incorrect index
- **IDXMOVE** — Moves indexes among application data storage areas
- **TABLEMOVE** — Moves tables among application data storage areas

For detailed descriptions of each VST, see Virtual System Tables on page 863.

Using the virtual system table mechanism, you can monitor the status and progress of several database administration utilities.

The following code sample demonstrates using an ABL routine to monitor all the index move (PROUTIL IDXMOVE) processes being performed:

```
/* Monitor all "index move" processes every three seconds */
REPEAT:
  FOR EACH _UserStatus WHERE _UserStatus-Operation = "Index move" NO-LOCK:
    DISPLAY _UserStatus.
  END.
  PAUSE 3.
END.
```

The following is an equivalent OpenEdge SQL statement to monitor all the index move (PROUTIL IDXMOVE) processes being performed:

```
SELECT * FROM _UserStatus WHERE _UserStatus-Operation = "Index move";
```

16

Multi-tenant database

The OpenEdge RDBMS includes multi-tenant support. Multi-tenancy allows an application provider to serve multiple customers (tenants) with one database. This chapter discusses multi-tenancy and changes in database administration due to multi-tenancy.

For an overview of multi-tenancy including new management tools, and language support, see *OpenEdge Getting Started: Identity Management*.

For details, see the following topics:

- Multi-tenant database introduction
- Multi-tenant database architecture
- Enabling multi-tenancy
- Creating multi-tenant elements
- Tenant Data Recovery
- Database utilities

Multi-tenant database introduction

A multi-tenant database is a database that provides support to a number of separate and distinct groups of users, referred to as tenants. Multi-tenant applications have taken an increasingly larger foothold in the database marketplace largely due to the emergence of SaaS as a deployment strategy. For the SaaS application provider a multi-tenant deployment strategy provides many benefits, especially when a shared tenancy approach is used. Management of the deployment can take place either at the overall application or database level, or based on tenant and user. Operational complexity is reduced when resources are shared and only one instance of the application and database needs to be maintained.

The process of developing and deploying multi-tenant applications is transformed and simplified by using the OpenEdge approach that reduces, and even eliminates, the challenges typically presented when implementing a shared tenancy application. This is due to two major factors:

- Tenant support is implemented in the database layer, completely removing the requirement that tenancy be managed by the application.
- Database utilities and tools are also tenant-aware.

For most applications, migrating an existing OpenEdge application to a multi-tenant application will only require minor changes to support the use of a multi-tenant database. The changes will be specific to the tenant authentication.

By implementing tenant support in the database layer, once the tenant identity is asserted the code that tenants run when using a multi-tenant database is exactly the same as with the same application running in a non-tenancy configuration. The tenancy (or non-tenancy) aspect is completely transparent. By updating the database utilities and tools to be database aware, the utilities can perform maintenance on tenant-specific data. For example, index maintenance, data dump and load, and object moves, can take place at the tenant level as well as across the entire database.

Database multi-tenancy is available to customers with an Enterprise RDBMS and an add-on product called OpenEdge Multi-tenant Tables, for deployment. For development, customers with a Development Server license can create and configure multi-tenant databases and develop multi-tenant applications.

Multi-tenant database architecture

A multi-tenant OpenEdge database is a shared database with a shared schema and logically and physically isolated data storage on a per tenant or group basis. Each object (table, index, LOB) is stored in a partition. Partitions keep data physically separate for each tenant. Partitions can exist in the same storage area or different storage areas. All multi-tenant objects are required to reside in a Type II area. If an object has an allocation state that indicates not to allocate data, then the partition is not created. Access to a partition can be disabled without deleting the data.

OpenEdge defines a tenant as a separate organizational entity within a multi-tenant database. For example, a tenant can equate to one company, or a division within one company. Access to a tenant's data is restricted to users that are authenticated to the tenant via a security domain. For information on security domains, see *OpenEdge Getting Started: Identity Management*.

For tables that are defined to be multi-tenant, each tenant has its own instance of the multi-tenant table (unless a partition for that table has not been allocated for that tenant). Tenants have access only to the data in the instances designated for that specific tenant as well as to tables that are shared. Two special cases are as follows:

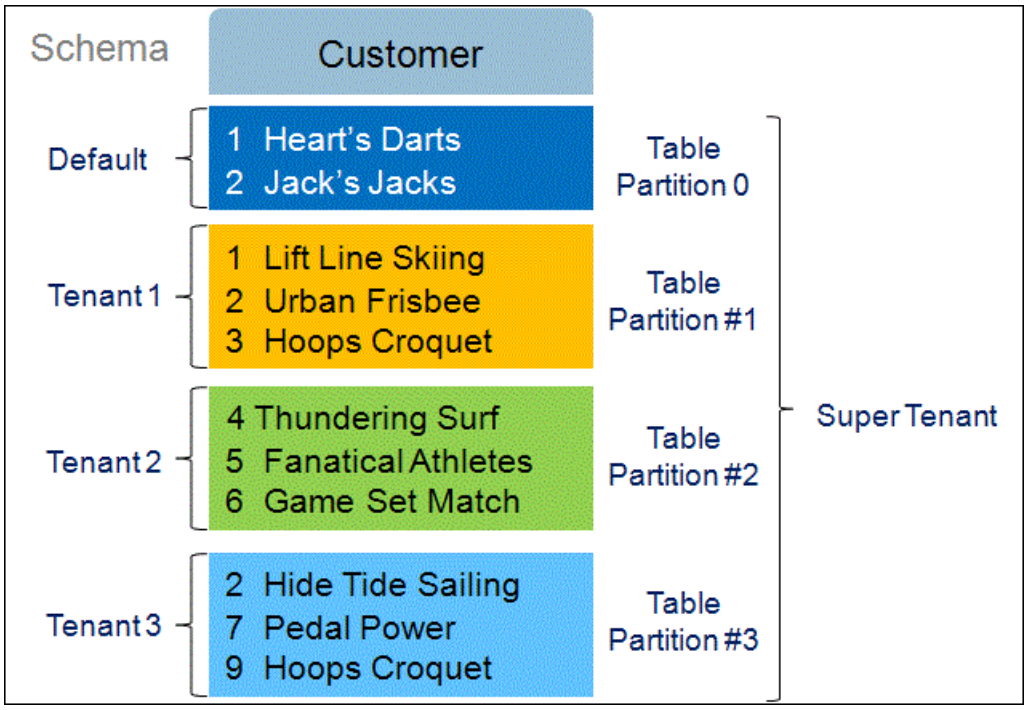
- **Super-tenants** — super-tenants can access all of the data in the database, typically for maintenance purposes.
- **Tenant groups** — Groups allow more than one tenant to access the same data for an instance of a table.

In a non-multi-tenant database, tables cannot span an area. In a multi-tenant database, each table partition can be stored in a separate area. Therefore, the maximum table size, limited by maximum area size, is controlled on a per-tenant basis. For more information on size limits, seeOpenEdge RDBMS Limits on page 59.

Simple tenancy

Multi-tenant tables have a 1-to-1 relationship between a tenant and a table partition. The following figure illustrates a multi-tenant table with one schema definition, and 3 tenants in addition to a default partition.

Figure 45: Simple tenancy example



The figure above illustrates that each tenant has its own table partition to maintain its data in. Each tenant can only see data associated with its own table partition. The super-tenant has access to all the table partitions of a multi-tenant table. When a tenant is deleted, its table partition is deleted.

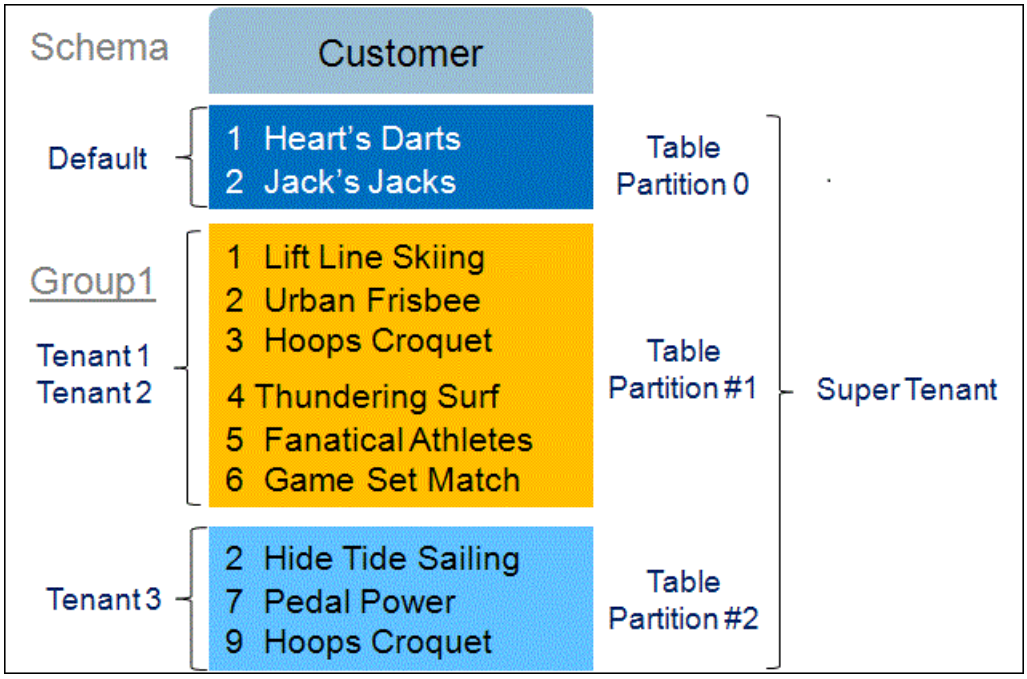
Multi-tenant groups

Multi-tenant groups allow multiple tenants to map to the same table partition for a multi-tenant table and therefore share the same data for that table. All data within the table partition is shared by the multiple tenants that are part of the group. Physical separation of the data by group, rather than by tenant.

Not all tenants need to participate in a multi-tenant group when a group exists on the table. A group is defined for one multi-tenant table. A tenant participating in a group for one multi-tenant table is not required to participate in a group for a different multi-tenant table. When deleting a tenant participating in a multi-tenant group, the table partition can only be deleted if it is not currently being shared by other tenants.

The following figure illustrates a multi-tenant table with a group. Tenant 1 and Tenant 2 belong to Group 1 and therefore share the same multi-tenant table partition and all data therein. Tenant 3 has its own table partition. While Tenant 1 and Tenant 2 have access to the same data, neither has access to Tenant 3's data, and Tenant 3 does not have access to Group 1's data.

Figure 46: Multi-tenant group



Multi-tenant features

An OpenEdge multi-tenant database delivers the following features:

- Tenant data is managed and maintained separately and independently.
- Data can be configured for physical separation per tenant (by specific area or by cluster within a specific area on a per multi-tenant object basis).
- Resources such as database sequences are viewed by the application as a single sequence definition but cam be maintained by the database on a per tenant basis.
- Analysis utilities are accessible globally as well as on a per tenant basis. For example, PROUTIL DBANALYS can be run across the database for all tenants or for a particular tenant.
- Activity monitoring through VSTs and PROMON is provided on a per tenant basis.
- Statistical data associated with specific objects for OpenEdge SQL query optimization is accumulated on a tenant/object specific manor.
- Tenancy is identified on a user connection basis, not on a process or server connection basis. This means that regardless of which process actually requests the data (AppServer, database server or direct connection) the data retrieved will be only for the authenticated tenant asserted.
- The addition of new tenants or the deletion of existing tenants will not affect the processing of other tenants of the same physical database.
- Tenants have access to non-multi-tenant (shared) tables.
- Users without tenancy have access to non-multi-tenant (shared) tables.

- There is one set of default values per table. All tenants share the same default values.
- Authorization is extended by the languages to the tenant level (rather than on the table level).
- The same `_user` name is supported across tenants. In other words, `joe_f` can be a user associated with both Tenant 1 and Tenant 2, and they are treated as different users with separate authentication.
- Column permissions are extended to be tenant specific.
- Super-tenants have the ability to query across tenants.

Enabling multi-tenancy

Prior to defining multi-tenant tables in your database, you must enable your database for multi-tenancy.

To enable an OpenEdge database for multi-tenancy, use:

```
proutil db-name -C enablemultitenancy
```

Once you have enabled multi-tenancy, you are ready to create tenants, groups, and multi-tenant tables, and begin the process of migrating your data so that is it configured for use by individual tenants.

Creating multi-tenant elements

Once you have a database enabled for multi-tenancy, you can begin to create and manage tenants, and structure their data. There are three ways to do this:

- Database administration tools on page 351
- ABL on page 352
- OpenEdge SQL on page 353

Database administration tools

Database administration tools that support multi-tenant configuration include the following:

- Database Administration Console on page 351
- Data Dictionary on page 352
- Data Administration on page 351

Database Administration Console

The easiest way to create and configure tenants and multi-tenant tables is with the **Database Administration Console** accessed through **OpenEdge Explorer** and **OpenEdge Management**. The **Data Administration Console** is the primary configuration tool for multi-tenancy. For complete details on using the console, see *OpenEdge Management and OpenEdge Explorer: Getting Started with Multi-tenancy* .

Data Administration

The multi-tenancy support in Data Administration ensures the following:

- Security Domains are defined as belonging to a Tenant in a multi-tenant database and that this information is included in dump and load.
- The built-in Security Authentication System and Authentication System Domain records are not included in data dump and the delete action and the fields that are not editable are disabled for these records in the user interface.
- Multi-tenant information is included when dumping and loading data definitions.
- A super-tenant can select and set the effective tenant when dumping and loading data and sequence current values and that the dump and load utilities have defaults that make it easy to dump and load tenant data to and from subdirectories named after the tenant. Dump Sequence Current Values dialog box
- Ensures that the maintenance of alternate buffer pools does not include multi-tenant tables.

For more information on Data Administration features, see the Data Administration online Help, or *OpenEdge Development: Basic Database Tools*.

ABL

The design of OpenEdge multi-tenant support enables ABL to automatically access tenant specific data without requiring the ABL programmer to syntactically specify the data they want to read or update by tenant. The user's tenancy is implicit in all ABL queries and data manipulation statements and the AVM will automatically confine the user's view of the data to the data segments for a particular tenant. This allows an existing application partner with few modifications to run their application in a multi-tenant database.

Multi-tenant ABL extensions have two main purposes:

- **API for multi-tenant maintenance** — ABL extensions that allow an ABL programmer to develop custom multi-tenant configuration tools.
- **ABL enhancements to support multi-tenant features** — enhancements that are focused on allowing super-tenant access.

For more information on the ABL API for multi-tenant maintenance, see the version of *OpenEdge Development: Programming Interfaces* on Progress Communities:https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx

Data Dictionary

The Data Dictionary multi-tenancy support makes it possible to edit and display multi-tenancy for tables and sequences, to edit and display whether a multi-tenant table has a default area, and display if the currently connected database is multi-tenant enabled.

Multi-tenant fields are added to all reports that show tables.

For more information on Data Dictionary features, see the Data Dictionary online Help, or *OpenEdge Development: Basic Database Tools*.

Data Administration

The multi-tenancy support in Data Administration ensures the following:

- Security Domains are defined as belonging to a Tenant in a multi-tenant database and that this information is included in dump and load.
- The built-in Security Authentication System and Authentication System Domain records are not included in data dump and the delete action and the fields that are not editable are disabled for these records in the user interface.
- Multi-tenant information is included when dumping and loading data definitions.

- A super-tenant can select and set the effective tenant when dumping and loading data and sequence current values and that the dump and load utilities have defaults that make it easy to dump and load tenant data to and from subdirectories named after the tenant. Dump Sequence Current Values dialog box
- Ensures that the maintenance of alternate buffer pools does not include multi-tenant tables.

For more information on Data Administration features, see the Data Administration online Help, or *OpenEdge Development: Basic Database Tools*.

ABL

The design of OpenEdge multi-tenant support enables ABL to automatically access tenant specific data without requiring the ABL programmer to syntactically specify the data they want to read or update by tenant. The user's tenancy is implicit in all ABL queries and data manipulation statements and the AVM will automatically confine the user's view of the data to the data segments for a particular tenant. This allows an existing application partner with few modifications to run their application in a multi-tenant database.

Multi-tenant ABL extensions have two main purposes:

- **API for multi-tenant maintenance** — ABL extensions that allow an ABL programmer to develop custom multi-tenant configuration tools.
- **ABL enhancements to support multi-tenant features** — enhancements that are focused on allowing super-tenant access.

For more information on the ABL API for multi-tenant maintenance, see the version of *OpenEdge Development: Programming Interfaces* on Progress Communities:https://community.progress.com/technicalusers/w/openedgegeneral/1329.openedge-product-documentation-overview.aspx

OpenEdge SQL

OpenEdge SQL delivers the following in language extensions to enable and support multi-tenancy:

- New multi-tenant specific SQL statements to create, alter, and drop tenants of an existing database table.
- Extended support for multi-tenancy in the OpenEdge SQL Data Definition Language (DDL) to create, alter, and drop multi-tenant enabled tables, indexes, and sequence.
- Extended support for multi-tenancy in the OpenEdge SQL Data Manipulation Language (DML) to select, insert, update, and delete to/from a multi-tenant table.

For more information on OpenEdge SQL syntax, see *OpenEdge Data Management: SQL Reference*.

Tenant Data Recovery

Tenant Data Recovery (TDR) is a process of ordered steps enabling database administrators to recover accidentally deleted table data and multi-tenant sequences for a tenant or group in a multi-tenant database without taking the entire database offline.

Once the need to restore data or a sequence is discovered, the high-level restoration process is as follows:

1. Locally restore your most recent backup of your production database to a temporary location.
2. Roll forward the after-image files of your production database to your temporary database if your production database is enabled for after-imaging.
3. Move data from the temporary database to your production database.
4. Clean up.

Figure 47: Tenant Data Recovery Process on page 355 illustrates the flow of the TDR process. The details of each step are discussed in the sections that follow.

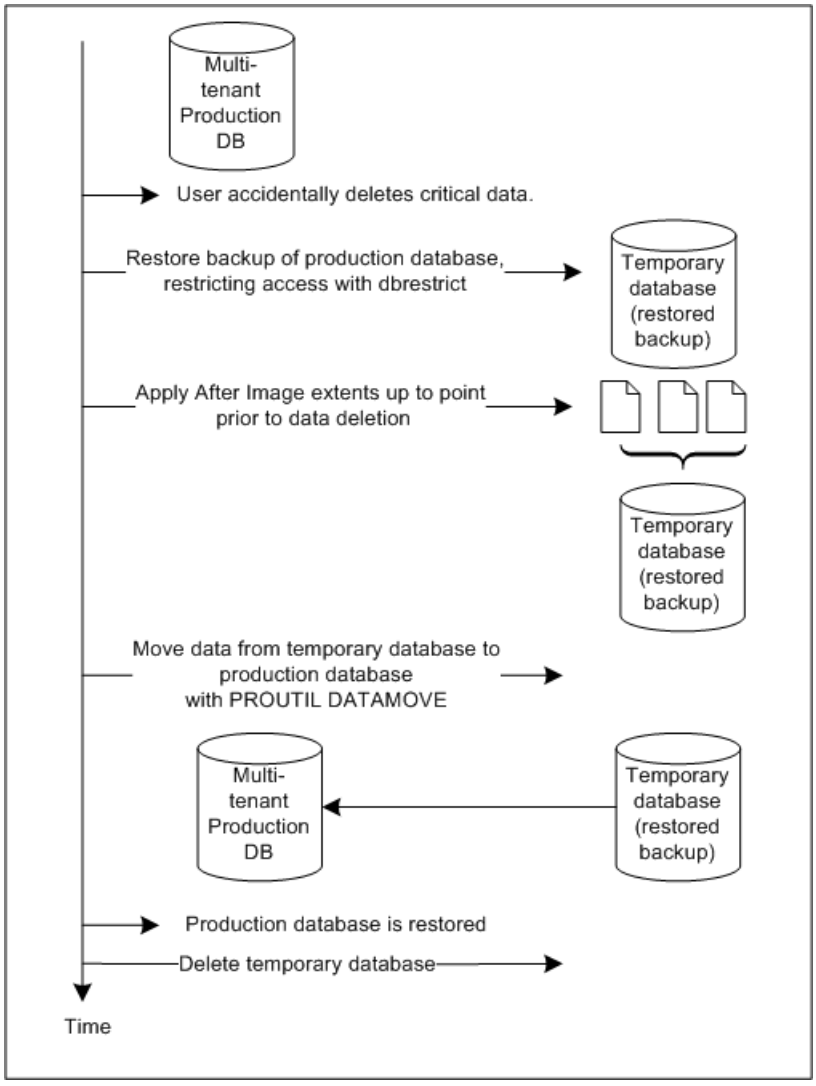
Restrictions

Restoring data with the TDR process has the following restrictions:

- Both the restored backup database and the target database must be enabled for multi-tenancy
- The restored backup database and target database must have identical schema for the table(s) or sequence(s) being moved
- The table partitions for tenants or groups must be allocated
- The tables must have indexes activated
- Dropped objects are not supported (only deleted data can be recovered, not table or sequence definitions)
- The restored backup database and the target database must be local (no remote connections)

- You can execute multiple data move commands simultaneously only if both databases are running in multi-user mode

Figure 47: Tenant Data Recovery Process



Step 1: Restoring your database backup

The first step in the tenant data recovery process is to restore your most recent backup of your production database to a temporary location.

When you restore your backup, you may want to protect the restored database from updates by other users. You achieve this by putting the database in a restricted access mode. You can restrict access to the database by adding the restriction parameter to the PROREST command. The syntax is:

```
prorest db-name . . . [ -dbrestrict restrict-type]
```

where *restrict-type* is one of the following values:

- **rollforward** — Indicating that the next allowable activity on this database is rolling forward after-image files

- **`datamove`** — Indicating that the next allowable activity on this database is moving data with the datamove utility

---

Note: All other `prorest` parameters are valid. Adding `-dbrestrict` to the `prorest` command enables the same restrictions as PROUTIL DBRESTRICT. See PROUTIL DBRESTRICT qualifier on page 675 for details of the restrictions.

The restriction you place on your database during the restore is determined by your use of after-imaging:

- If you use after-imaging, your next step in the TDR process is applying after-image files to your restored database. Choose the value `rollforward` for *`restrict-type`*.
- If you don't use after-imaging, your next step in the TDR process is executing the PROUTIL DATAMOVE. Choose `datamove` as the value for *`restrict-type`*.

Restricting access to the restored database ensures that no changes can be made to the database before the TDR process completes. If you don't restrict access, a user could connect and make changes, resulting in the TDR process restoring incorrect data to your production database.

Step 1A: restricting access to your temporary database

If you don't restrict access to the database with the PROREST command, you can restrict access with the PROUTIL DBRESTRICT command as a separate step. The syntax is:

```
proutil db-name -C dbrestrict restrict-type restrict-action
```

For example, to restrict access for moving data on a database named `temporary_db`, you would enter:

```
proutil temporary_db -C dbrestrict datamove enable
```

For the complete syntax of PROUTIL DBRESTRICT, see PROUTIL DBRESTRICT qualifier on page 675.

Step 2: Rolling forward after-image files

The second step in the tenant data recovery process is to apply after-image files to your temporary database.

After restoring your production database backup to a temporary database, if you use after-imaging, you need to apply all the after-image files from the production database that have been created since the backup was created, up to the point just prior to when the accidental deletion occurred. If you don't use after-imaging, proceed to Step 3: Moving data with PROUTIL DATAMOVE on page 358.

To expedite the process of rolling forward multiple after-image extents, RFUTIL ROLL FORWARD accepts a file containing a list of after-image files as input. The syntax of ROLL FORWARD with a list of extents is:

```
rfutil db-name -C roll forward -ailist listfile
```

The *`listfile`* file, containing a list of AI area files, must be structured as follows:

- Specify one AI file per line

- AI file names must have a fully specified absolute file name or a relative file name, relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

If a file in the list is not an after-image file, ROLL FORWARD issues a warning and the file is skipped.

The following example demonstrates applying the after image files from a database named `production_db` to a database named `temporary_db`.

To apply after-image files from a list:

1. Create a list of after-image files from `production_db`:

```
proenv>ls production_db.a* > production_ai_files.lst

proenv>cat production_ai_files.lst
production_db.a1
production_db.a10
production_db.a11
production_db.a12
production_db.a2
production_db.a3
production_db.a4
production_db.a5
production_db.a6
production_db.a7
production_db.a8
production_db.a9
```

   Notice that the files are sorted using an alphabetical sort. ROLL FORWARD sorts the files into the proper sequence order before applying them to the target database.

2. Apply the source after-image files to the target, `temporary_db`, as shown:

```
proenv>rfutil temporary_db -C roll forward -ailist
          production_ai_files.lst
```

   ROLL FORWARD sorts and applies the after-image files, reporting the progress of each file as it is applied to both the screen and the database log file. The following is a portion of the output:

```
OpenEdge Release 11.3ALPHA as of Wed Mar 13 19:01:40 EDT 2013
***Beginning processing of After-image extent C:\OE113WRK\production_db.a1 (17047)
After-image dates for this after-image file: (1633)
    Last AIMAGE BEGIN Mon Mar 18 18:12:06 2013 (1640)
    This is aimage file number 1 since the last AIMAGE BEGIN. (1642)
    This file was last opened for output on Mon Mar 18 18:12:11 2013. (1643)

10% of aimage file processed (964 notes applied)...
20% of aimage file processed (1924 notes applied)...
30% of aimage file processed (2751 notes applied)...
40% of aimage file processed (3713 notes applied)...
50% of aimage file processed (4628 notes applied)...
60% of aimage file processed (5587 notes applied)...
70% of aimage file processed (6550 notes applied)...
80% of aimage file processed (7466 notes applied)...
90% of aimage file processed (8424 notes applied)...

8932 notes were processed. (1634)
```

```
0 in-flight transactions. (3785)
1 transactions were started. (1635)
1 transactions were completed. (11138)

At the end of the .ai file, 0 transactions were still active. (1636)
***Finished processing of After-image extent C:\OE113WRK\production_db.a1 (17048)

***Beginning processing of After-image extent C:\OE113WRK\production_db.a2 (17047)
After-image dates for this after-image file: (1633)
    Last AIMAGE BEGIN Mon Mar 18 18:12:06 2013 (1640)
    Last AIMAGE NEW Mon Mar 18 18:12:58 2013 (1641)
    This is aimage file number 2 since the last AIMAGE BEGIN. (1642)
    This file was last opened for output on Mon Mar 18 18:12:58 2013. (1643)
..
***Finished processing of After-image extent C:\OE113WRK\production_db.a11 (17048)

***Beginning processing of After-image extent C:\OE113WRK\production_db.a12 (17047)
After-image dates for this after-image file: (1633)
    Last AIMAGE BEGIN Mon Mar 18 18:12:06 2013 (1640)
    Last AIMAGE NEW Mon Mar 18 18:25:26 2013 (1641)
    This is aimage file number 12 since the last AIMAGE BEGIN. (1642)
    This file was last opened for output on Mon Mar 18 18:25:26 2013. (1643)

108977 notes were processed. (1634)
1 in-flight transactions. (3785)
0 transactions were started. (1635)
1 transactions were completed. (11138)
At the end of the .ai file, 0 transactions were still active. (1636)
Roll forward of the after-image busy extent is complete. (3786)
All roll forward for database target_db is complete. (3787)
***Finished processing of After-image extent C:\OE113WRK\production_db.a12 (17048)
```

Step 3: Moving data with PROUTIL DATAMOVE

The third step in the tenant data recovery process is to move data from your temporary database to your production database.

Once you have a temporary database restored from backup and rolled forward, you can use it as the source database for recovering your deleted data. You move data into your production database with PROUTIL DATAMOVE. You can issue a complete DATAMOVE from the command line, or you can invoke the menu-driven user interface, provided the target and source databases are running in multi-user mode.

A single command line data move allows you to move data for one tenant or group. For a tenant, you can move table data or a sequence; for a group, you can move table data.

Depending on the data you need to recover, choose one or more of the following options to PROUTIL DATAMOVE:

- Move one table with the DATAMOVE command line on page 359
- Move one sequence with the DATAMOVE command line on page 359
- Move multiple tables and sequences with the DATAMOVE user interface on page 359

By default, PROUTIL DATAMOVE, moves data by bundling 100 records in a single transaction. You can change the number of records per transaction on the command line with the `recs` parameter, or with Option 4 of the user interface.

See PROUTIL DBRESTRICT qualifier on page 675 for the complete syntax.

Move one table with the DATAMOVE command line

To move one table for a tenant with the DATAMOVE command line, the syntax is as follows:

```
proutil target-database -C datamove source source-database tenant tenant-name table
  table-name
```

For example, if you want to restore data to a production database named `production_db`, for the `customer` table for a tenant named `store1`, and your source database is named `temporary_db`, enter the following command:

```
proutil production_db -C datamove source temporary_db tenant store1
     table customer
```

Similar to moving data for a tenant, you can move data for a group with the following syntax:

```
proutil target-database -C datamove source source-database group group-name table
  table-name
```

Move one sequence with the DATAMOVE command line

To move one sequence from the DATAMOVE command line, the syntax is as follows:

```
proutil target-database -C datamove source source-database tenant tenant-name

     sequence sequence-name
```

For example, if you want to restore data to a production database named `production_db`, for the sequence `customer_id` for a tenant named `store1`, and your restored source database is named `temporary_db`, enter the following command:

```
proutil production_db -C datamove source temporary_db tenant store1
     sequence customer_id
```

Move multiple tables and sequences with the DATAMOVE user interface

You can invoke the DATAMOVE user interface to make multiple selections of either tables or sequences to move. In order to invoke the user interface, both your source and target databases must be started in multi-user mode. Invoke the command line using the following syntax:

```
proutil target-database -C datamove source source-database
```

For example, if your target database is named `production_db`, and your source database is named, `temporary_db`, start the user interface with the following command:

```
proutil production_db -C datamove source temporary_db
```

The user interface menu appears as shown:

```
 Progress OpenEdge Tenant/Group Data Move

    Source Database     : temporary_db
    Destination Database: production_db

    1. Move Table Data by Tenant
    2. Move Sequence Data by Tenant
    3. Move Table Data by Group
    4. Set Records per Transaction ( Current: 100 )
    5. Quit

Enter your selection: q
```

From this menu, select an option to proceed. The steps are explained in the sections that follow.

Move table data by tenant

When you select Option 1 of Tenant/Group Data Move, you can move data from multiple tables for one tenant.

Use the following steps to move data by tenant:

1. From the main menu, choose **1**.

   The following screen appears:

```
                Please Choose a Tenant
------------------------------------------------------------------------
Option|       Tenant Name       |  ID  |       Description
------------------------------------------------------------------------
     1|Default                  |     0|Default Tenant
     2|store1                   |     1|
     3|store2                   |     2|
     4|store3                   |     3|
     5|store4                   |     8|
     6|store5                   |     9|
     7|allocdis                 |     4|
     8|delayalloc               |     7|
     9|noalloc                  |     5|
    10|noallocdis               |     6|
------------------------------------------------------------------------
(Page 1 of 1)
------------------------------------------------------------------------
                             Input Your Choice
 (c/C) - Change Display
     (q/Q) - Quit   (g/G) - Go
```

   Note: Observe that tenants that have delayed allocation or no allocation are included in the list.

2. Choose a Tenant by entering a number and pressing return.

For example, if you enter **3**, to select the tenant named `store2`, and an asterisk appears next to your choice:

```
                Please Choose a Tenant
------------------------------------------------------------------------
Option|       Tenant Name       |  ID  |       Description
------------------------------------------------------------------------
     1|Default                  |     0|Default Tenant
     2|store1                   |     1|
*    3|store2                   |     2|
     4|store3                   |     3|
     5|store4                   |     8|
     6|store5                   |     9|
     7|allocdis                 |     4|
     8|delayalloc               |     7|
     9|noalloc                  |     5|
    10|noallocdis               |     6|
------------------------------------------------------------------------
(Page 1 of 1)
------------------------------------------------------------------------
                             Input Your Choice
 (c/C) - Change Display
     (q/Q) - Quit   (g/G) - Go
```

Note: To undo any selection, select the item a second time.

3. Once you have selected your tenant, enter **g** or **G** for go.

   The following menu appears for you to choose a table or tables:

```
                Choose Tables for Data Move
------------------------------------------------------------------------
Option|          Table Name         |     Table Owner          |Number
------------------------------------------------------------------------
     1|Customer                     |PUB                       |   10
     2|Employee                     |PUB                       |   12
     3|Invoice                      |PUB                       |   14
     4|Stock                        |PUB                       |   16
     5|Vendor                       |PUB                       |   18
------------------------------------------------------------------------
Page 1 of 1
                             Input Your Choice
 (c/C) - Change Display
 (a/A) - All     (q/Q) - Quit   (g/G) - Go
```

4. Select the table or tables you want to move data for by entering the number from the Option column.

   For each table you select, an asterisk appears.

For example, after choosing options **1**, **3**, and **5**, the menu appears as follows:

```
                Choose Tables for Data Move
------------------------------------------------------------------------
Option|          Table Name          |     Table Owner          |Number
------------------------------------------------------------------------
*    1|Customer                      |PUB                       |   10
     2|Employee                      |PUB                       |   12
*    3|Invoice                       |PUB                       |   14
     4|Stock                         |PUB                       |   16
*    5|Vendor                        |PUB                       |   18
------------------------------------------------------------------------
Page 1 of 1
                             Input Your Choice
 (c/C) - Change Display
 (a/A) - All     (q/Q) - Quit   (g/G) - Go
```

5. Enter **g** or **G** for go, and begin the data move. DATAMOVE provides status updates during the moving of data.

Note: To cancel the move, enter CTRL-C.

Move Sequence values by tenant

When you select Option 2 of Tenant/Group Data Move, you can move data from multiple sequences for one tenant.

Use the following steps to move sequences by tenant:

1. From the main menu, choose **2**.

   The following screen appears:

```
                Please Choose a Tenant
------------------------------------------------------------------------
Option|       Tenant Name       |  ID  |       Description
------------------------------------------------------------------------
     1|Default                  |     0|Default Tenant
     2|store1                   |     1|
     3|store2                   |     2|
     4|store3                   |     3|
     5|store4                   |     8|
     6|store5                   |     9|
     7|allocdis                 |     4|
     8|delayalloc               |     7|
     9|noalloc                  |     5|
    10|noallocdis               |     6|
------------------------------------------------------------------------
(Page 1 of 1)
------------------------------------------------------------------------
                             Input Your Choice
 (c/C) - Change Display
     (q/Q) - Quit   (g/G) - Go
```

2. Choose a Tenant by entering a number and pressing return.

For example, if you choose **6**, for the tenant named `store5`, an asterisk appears next to your choice, as shown:

```
                Please Choose a Tenant
------------------------------------------------------------------------
Option|       Tenant Name       |  ID  |       Description
------------------------------------------------------------------------
     1|Default                  |     0|Default Tenant
     2|store1                   |     1|
     3|store2                   |     2|
     4|store3                   |     3|
     5|store4                   |     8|
*    6|store5                   |     9|
     7|allocdis                 |     4|
     8|delayalloc               |     7|
     9|noalloc                  |     5|
    10|noallocdis               |     6|
------------------------------------------------------------------------
(Page 1 of 1)
------------------------------------------------------------------------
                             Input Your Choice
 (c/C) - Change Display
     (q/Q) - Quit   (g/G) - Go
```

Note: To undo any selection, select the item a second time.

3. Once you have selected your tenant, enter **g** or **G** for go.

   The following menu appears for you to choose a sequence:

```
   Choose Sequences for Data Move for Tenant: store5
------------------------------------------------------------------------
Option|         Sequence Name          |  Number
------------------------------------------------------------------------
     1|CustomerID                      |       1
     2|VendorID                        |      10
     3|EmployeeID                      |       2
     4|InvoiceNum                      |       3
     5|StockNum                        |       4
     6|OrderNum                        |       5
     7|RefNum                          |       6
     8|SaleNum                         |       7
     9|ShippingID                      |       8
    10|MgrID                           |       9
    11|SEQ1                            |       0
------------------------------------------------------------------------
Page 1 of 1
                             Input Your Choice
 (c/C) - Change Display
 (a/A) - All     (q/Q) - Quit   (g/G) - Go
```

4. Select the sequence or sequences you want to move data for by entering the number from the Option column.

For each sequence you select, an asterisk appears. For example, after choosing options **6**, **7**, and **8**, the menu appears as follows:

```
   Choose Sequences for Data Move for Tenant: store5
------------------------------------------------------------------------
Option|         Sequence Name          |  Number
------------------------------------------------------------------------
     1|CustomerID                      |       1
     2|VendorID                        |      10
     3|EmployeeID                      |       2
     4|InvoiceNum                      |       3
     5|StockNum                        |       4
*    6|OrderNum                        |       5
*    7|RefNum                          |       6
*    8|SaleNum                         |       7
     9|ShippingID                      |       8
    10|MgrID                           |       9
    11|SEQ1                            |       0
------------------------------------------------------------------------
Page 1 of 1
                              Input Your Choice
 (c/C) - Change Display
 (a/A) - All     (q/Q) - Quit   (g/G) - Go
```

5. Enter **g** or **G** for go, and begin the data move. DATAMOVE provides status updates during the moving of data.

Note: To cancel the move, enter CTRL-C.

Move table data by group

When you select Option 3 of Tenant/Group Data Move, you can move data from multiple tables for one group.

Use the following steps to move data by group:

1. From the main menu, choose **3**.

   The following screen appears:

```
 Please Choose a Group
------------------------------------------------------------------------
Option|          Group Name            |  ID  |          Description
------------------------------------------------------------------------
     1|NEregion                        |    -1|
     2|SEregion                        |    -2|
     3|SWregion                        |    -3|
     4|NWregion                        |    -4|
------------------------------------------------------------------------
(Page 1 of 1)
------------------------------------------------------------------------
                              Input Your Choice
 (c/C) - Change Display
     (q/Q) - Quit   (g/G) - Go
```

2. Choose a group by entering a number and pressing return.

For example, if you enter **3**, to select the tenant named `SWregion`, and an asterisk appears next to your choice:

```
 Please Choose a Group
------------------------------------------------------------------------
Option|          Group Name            |  ID  |          Description
------------------------------------------------------------------------
     1|NEregion                        |    -1|
     2|SEregion                        |    -2|
*    3|SWregion                        |    -3|
     4|NWregion                        |    -4|
------------------------------------------------------------------------
(Page 1 of 1)
------------------------------------------------------------------------
                             Input Your Choice
 (c/C) - Change Display
     (q/Q) - Quit   (g/G) - Go
```

Note: To undo any selection, select the item a second time.

3. Once you have selected your group, enter **g** or **G** for go.

   The following menu appears for you to choose a table or tables:

```
 Choose Tables for Data Move
------------------------------------------------------------------------
Option|          Table Name            |          Table Owner           |Number
------------------------------------------------------------------------
     1|Employee                       |PUB                             |   12
     2|Invoice                        |PUB                             |   13
------------------------------------------------------------------------
Page 1 of 1
                             Input Your Choice
 (c/C) - Change Display
 (a/A) - All     (q/Q) - Quit   (g/G) - Go
```

4. Select the table or tables you want to move data for by entering the number from the Option column.

   For each table you select, an asterisk appears. For example, after choosing option **1**, the menu appears as follows:

```
 Choose Tables for Data Move
------------------------------------------------------------------------
Option|          Table Name            |          Table Owner           |Number
------------------------------------------------------------------------
*    1|Employee                       |PUB                             |   12
     2|Invoice                        |PUB                             |   13
------------------------------------------------------------------------
Page 1 of 1
                             Input Your Choice
 (c/C) - Change Display
 (a/A) - All     (q/Q) - Quit   (g/G) - Go
```

5. Enter **g** or **G** for go, and begin the data move. DATAMOVE provides status updates during the moving of data.

Note: To cancel the move, enter CTRL-C.

Set records per transaction

When performing a data move, a specified number of records are bundled into one transaction. By default, the number of records is 100. You can change the number.

To change the number of records for each data move transaction:

1. From the main menu, choose **4**

   The following line appears:

```
Current Records per Transaction is: 100, Input Your Choice: ( > 1)
```

2. Enter a positive integer.

Step 4: Tenant data recovery clean up

Once you have completed moving data with DATAMOVE, you can remove the access restriction from your temporary (DATAMOVE source) database and delete it.

Remove access restriction from your source database

Before executing DATAMOVE, you restricted access to the source database (temporary restored backup). Access continues to be restricted until you disable the restriction.

To remove the access restrictions from your source database:

1. If you started a server for source database, shut down the server.
2. Once the server is shut down, query the status of the datamove restriction with PROUTIL DBRESTRICT.

   For example, if the source database is named `temporary_db`, enter the following:

```
proutil temporary_db -C dbrestrict datamove status
```

   If the database is restricted, the following is displayed:

```
Restricted Access Mode is enabled for DataMove. (17073)
```

3. Remove the restriction with PROUTIL DBRESTRICT.

Entering the following command:

```
proutil temporary_db -C dbrestrict datamove disable
```

The results of removing the restriction is as follows:

```
Database Restricted Access mode has been disabled for DataMove (17039)
```

Delete source database

Once you have completely restored your production database with the TDR process, your temporary restored backup is no longer needed. You can delete this database.

Clear state information from your target database

The `clear all` option removes all DATAMOVE state information from your database. The `clear all` option is necessary only in the rare cases that your database needs to be used with a version of OpenEdge earlier than Release 11.3.0 and a DATAMOVE operation terminated prior to completion.

To remove any indicators of an incomplete state, and to disable the ability to continue to executing PROUTIL DATAMOVE on the database, use the `clear all` parameter to DATAMOVE. Invoke the command line using the following syntax:

Syntax

```
proutil target-database -C datamove clear all
```

For example, to clear a database named `production_db`, enter the following:

```
proutil production_db -C datamove clear all
```

The results of executing a clear all are as follows:

```
Executing to clear all objects. (16960)
The feature Data Move has been successfully disabled. (16965)
Clear all objects completed successfully. (16962)
```

Database utilities

Database utility changes in support of multi-tenancy fall into four categories:

- Unchanged utilities on page 368
- Indirectly changed utilities on page 368
- Multi-tenant aware utilities on page 368

- Controlling the usage of multi-tenant database resources using multi-tenant governors on page 370

Unchanged utilities

Unchanged database utilities continue to operate across the database, and are not impacted by multi-tenancy. These utilities include:

- Backup and restore
- After-imaging and Replication
- Failover clusters
- Prostrct
- Transparent Data Encryption

Indirectly changed utilities

Auditing is a utility that is indirectly changed. The mechanics of how auditing is managed remains unchanged, but auditing has the following changes:

- New audit policies and auditing events are added and extended to support multi-tenancy. An incremental audit policy, `multi-tenant.xml`, is included in the `%DLC%/auditing` directory. The full auditing policy will also have the incremental one appended to it.
- Audit event triggers are included in the implementation of the new enable and disable multi-tenant feature commands in PROUTIL.
- Existing database record auditing event triggers are updated to include the tenant-name information.

Multi-tenant aware utilities

Tenant aware utilities allow the database administrator to execute the utilities on a per-tenant, per-group, or database-wide basis. Database utilities that can manipulate and display data based on multi-tenant characteristics are added or updated. For the complete syntax of each utility, see the corresponding chapter in Part 4, "Reference." The list impacted utilities follows:

- PROUTIL specific to multi-tenant databases commands
  - ENABLEMULTITENANCY
  - DISABLEMULTITENANCY
  - DEALLOCATE
  - DATAMOVE (See Tenant Data Recovery on page 353)
  - MTIDXBUILD (See Multi-tenant index rebuild on page 369)
- Modified PROUTIL commands:
  - BULKLOAD
  - DBANALYS/IDXANALYS/TABANALYS
  - DESCRIBE
  - DISPTOSSCREATELIMITS

- DUMP/DUMPSPECIFIED
- IDXACTIVATE
- IDXBUILD/IDXCHECK/IDXFIX
- IDXDEALLOCATE
- IDXMOVE
- LOAD
- Create and Toss limits (SETAREACREATELIMIT, SETAREATOSSLIMIT, SETLOBCREATELIMIT,SETLOBTOSSLIMIT,SETTABLECREATELIMIT,SETTABLETOSSLIMIT)
- TABLEMOVE
- VIEWB2

- DBTOOL
- SQLDUMP/SQLLOAD/SQLSCHEMA
- PROMON

The OpenEdge RDBMS and utilities fully support the authentication and authorization enhancements that are part of OpenEdge Identity Management. The utilities accept the -U/-P parameters. If no username and password are included, the utilities trust the underlying operating system authentication and authorization. For more information on Identity Management, see *OpenEdge Getting Started: Identity Management*.

Multi-tenant index rebuild

For databases enabled for multi-tenancy, the MTIDXBUILD utility allows database administrators to rebuild an index or indexes for a specific tenant or group while the database remains online, in addition to offline. MTIDXBUILD only rebuilds indexes for multi-tenant tables.

To rebuild a multi-tenant index, follow this single-step process:

1. Perform the index rebuild using the PROUTIL MTIDXBUILD command (online or offline). See PROUTIL MTIDXBUILD qualifier on page 743.

Multi-tenant index rebuild acquires an exclusive table partition lock when performing the rebuild, blocking other processes from creating or deleting index objects. Once the rebuild starts, clients cannot use the index in a query, locked or no-lock.

To prevent users from accessing a rebuilt index with a stale cursor, ABL clients, whose login tenant or group matches the table MTIDXBUILD is working on, or whose login id is a super tenant, with a timestamp older than a current timestamp from MTIDXBUILD, must log out.

Multi-tenant index rebuild has significant differences from the full index rebuild (PROUTIL IDXBUILD). Those differences include:

- Multi-tenant index rebuild can operate while the database is online.
- You can still access the database if the index rebuild fails. If the index rebuild fails, the index is left offline and must be brought online with an index utility (index rebuild or activation) because not all of the index rebuild activity is logged. Users can access the database after a failed multi-tenant index rebuild, provided the new process goes through crash recovery.
- You cannot use multi-tenant index rebuild to clear a tainted index flag. Only the full index rebuild can do that.

- Multi-tenant index rebuild is not multi-threaded.
- To improve performance, multi-tenant index rebuild deletes and rebuilds the index.
- There is no menu-driven option to MTIDXBUILD, you must completely specify the rebuild on the command line.

You can track the progress of MTIDXBUILD using the User status (`_UserStatus`) Virtual System Table.

Controlling the usage of multi-tenant database resources using multi-tenant governors

There are two multi-tenant governors which you can use to control the usage of multi-tenant database resources. These governors limit a tenant's access to the shared database resources. These multi-tenant governors are:

- Login Governor (`-nGovernor`)
- Lock Governor (`-LGovernor`)

Note: These governors are applicable to only the default tenant and regular tenant in multi-tenant databases; these governors do not apply to a super-tenant. Also, these governors do not apply to non multi-tenant databases.

The Login Governor prevents a tenant from using all the database connections in a multi-tenant database. The Lock Governor prevents a tenant from acquiring all the locks in a lock table. You can regulate the amount of multi-tenant database resources used by a tenant by enabling these database startup parameters: `-nGovernor` and `-LGovernor`.

The Startup Virtual System Table (VST) contains the fields for the `-nGovernor` and `-LGovernor` startup parameters. For more information on Startup VST, see .

The PROMON utility and Database Administration console help you to monitor and change the governor values for these two startup parameters. For more information on how the PROMON utility monitors online Lock Governor, see PROMON Utility on page 543. For more information about Database Administration console, see *OpenEdge Management and OpenEdge Explorer: Configuring Multi-tenancy*.

When a governor is enabled in an online database, shared memory is allocated for the governor data structure. If the shared memory becomes insufficient, use the PROUTIL INCREASETO command to provide at least 256KB of additional shared memory. See PROUTIL INCREASETO qualifier on page 736.

17

Table-partitioned database

The OpenEdge RDBMS includes table partitioning support. Table partitioning allows database administrators and application developers to design databases that partition data storage to improve data access and concurrency. This chapter discusses table partitioning and changes in database administration due to table partitioning.

For an overview of table partitioning including new management tools, and language support, see *OpenEdge Getting Started: Table Partitioning*.

For details, see the following topics:

- Table partitioning details
- Table partition types
- Table partitioning restrictions
- Indexes for table partitions
- Partitioning your database
- Managing partitions
- Partition-level database administration

Table partitioning details

Table partitioning enables one table to be divided into multiple self-contained sub-sections (partitions) that can be located in different physical areas. A row from a partitioned table never spans a partition boundary. Prior to OpenEdge 11.4, a table, regardless of size, was physically stored in only one storage area. It was also treated as a single unit at the application and database levels. Consequently, for a very large table, it could be very slow and costly to perform certain operations, either run-time queries or maintenance activities.

Partitioning tables can provide advantages, including the following:

- Improved performance
  - If querying with a partition-aligned index, whole partitions can be quickly removed from consideration. This is called pruning.
  - Concurrency is increased in random activity.
- Improved availability

  Maintenance can be performed on a single partition, leaving other partitions from a table available.
- Improved maintenance

  Utilities can operate on one partition, reducing the amount of time that data is unavailable (locked by the utility). Partition maintenance includes partition management to split, merge and purge partitions; partition-level index management such as index check, fix and rebuild; and partition-level data activities such as binary dump and load.

Partitioning tables also has the risk associated with these disadvantages:

- Correct partitioning requires knowledge of table layout and application deign to be aligned with the physical layout of the database. Re-partitioning is expensive, requiring a dump and load of data.
- Updating a partition-aligned column can cause a row to be moved from one partition to another.

Table partition types

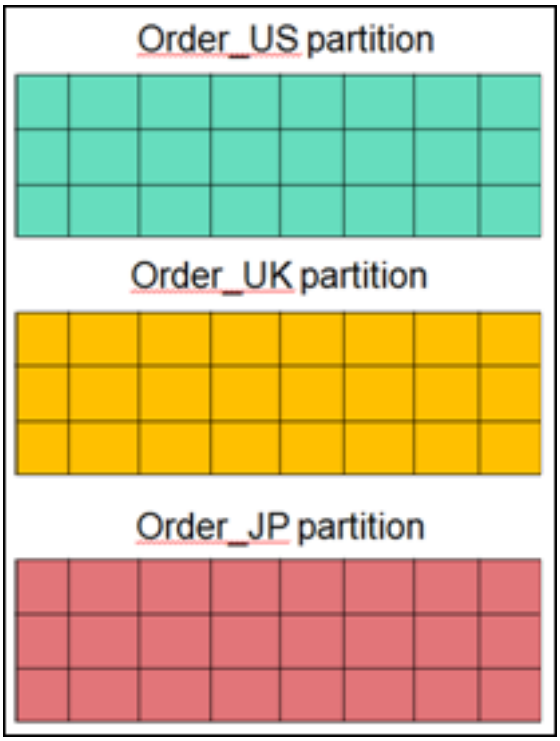
Table partitioning enables you to divide the rows of large tables into smaller units, called partitions, and manage each partition individually. The table appears as one single unit to the application, but it is physically divided into multiple partitions in the database. OpenEdge table partitioning supports several different kinds of partitions, as follows:

- List

  For a list partition, the partition definition is based on a single value, and the values for the column that define the partition define a complete set. You cannot create a row that does not match the partition definition.

For example, suppose you had a table of orders containing a country column, and you wanted to partition the column based on the three possible country values, JP, UK and US, the list-partitioned table would look like the following figure:

Figure 48: Example list partition

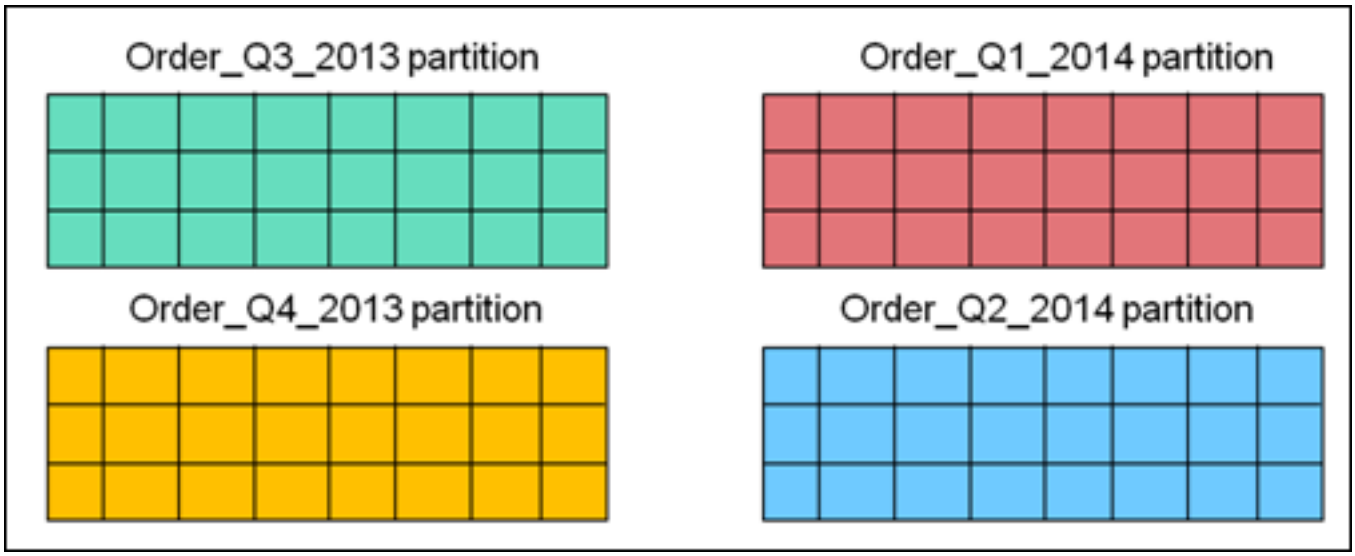


- Range

  For a range partition, values for the column that define the partition match particular ranges. Ranges cannot overlap. For example, if you are partitioning based on a date field and defining partitions for a year's worth of data, your partitions define ranges of January 1 through December 31, for each year.

  For example, suppose you had a table of orders containing an order date column, and you wanted to partition the column based on quarters of the calendar year. The range-partitioned table would look like the following figure:

Figure 49: Example range partition



- Sub-partitions

  You can partition a partition, to further refine the subset of a table.

  - **List-range** — A list-range partition first defines a partition by a list, and then further partitions the data based on a range.
  - **List-list** — A list-list partition first defines a partition by a list, and then further partitions the data based on a subsequent list.

  You can create up to 15 levels of sub-partitions. You can have 15 list sub-partitions, or you can have 14 list sub-partitions, and finally a last sub-partition a range partition.

  For example, taking the two previous examples of a list partition and a range partition on a table of orders, combining them into a list-range partitioned table would look like the following figure:

Figure 50: Example list-range partition



- Composite

  A composite partition is a partition that contains the data for multiple partition definitions in the same physical partition. Intended as a migration tool, when initially partitioning an existing table, all data resides in the *composite initial* partition.

Table partitioning restrictions

Table partitioning restrictions include the following:

- Table partitions must reside in Type II areas.
- Partitions can not span storage areas.
- A maximum of 32,768 partitions (one of which is the composite initial partition) are allowed per table.
- A maximum of 15 sub-partitions are allowed per table.
- Only one range column per table may be specified as defining a range partition; it can be either the only partition or the last sub-partition.

- Overlapping range partitions are not supported. Ranges are contiguous, without gaps. Data in each range partition is all data less than or equal to (<=) the partition definition value down to the previous partition definition or down to the beginning of time if there is no smaller value.
- Partition-aligned columns in a local index must be ascending.
- Once you partition a table, if you need to re-define partitions for that table, you must dump and load the data.
- Once you enable table partitioning for your database, you must remove all partitions to in order to disable the feature.
- Table partitioning is not supported for multi-tenant tables.

Note: Note you can have both multi-tenant and partitioned tables in the same database, but you cannot use both features on the same table.

Indexes for table partitions

Two types of indexes support locating data within a partitioned table, as follows:

- Global

  A Global index on a partitioned table spans all partitions. There is one index b-tree for all of the data, and provides uniqueness for non-partitioned columns.

- Partition-aligned local

  A partition-aligned local index has one index b-tree for each partition.

Note: Word indexes can not be local indexes.

Partitioning your database

With a high-level understanding of table partitioning, you can begin the process of partitioning your database. While a specific process is unique to every situation, the following is a high-level process outline:

- Understanding what tables you will partition.

  You will need to evaluate the layout and content of your database to determine the best tables to partition and how they are partitioned (list, range, sub-partition). Consult *OpenEdge Getting Started: Table Partitioning* for a discussion of the factors you should consider when planning your partitioning.

- Understanding your physical layout.

  You will need to evaluate the current storage layout of your database. As you partition your database, do you need new storage areas to hold your new partitions? Do you want to move tables and indexes from their existing storage areas? Use the existing PROSTRCT and PROUTIL tools to make these changes.

- Enabling partitioning.

  Prior to creating partitions, you will need to enable your database. You can use PROUTIL or OpenEdge Management/Explorer to enable your database.

- Defining partitions

  Once you have a plan, you have several options to define your partitions:

  - OpenEdge Management/Explorer provides a graphical interface to define partitions. See *Getting Started: OpenEdge Management and OpenEdge Explorer* and *OpenEdge Explorer: Managing Table Partitioning in Databases*.
  - OpenEdge SQL provides DDL for partitioning. See *OpenEdge Data Management: SQL Development* and *OpenEdge Data Management: SQL Reference* for details.
  - OpenEdge ABL provides APIs for partitioning. See *OpenEdge Development: Programming Interfaces* for details.

- Migrating data

  When you first create partitions on an existing table, the data remains in the composite initial partition. You can use the PROUTIL PARTITIONMANAGE SPLIT option to move data out of the initial partition and into other partitions. If you have a new partitioned table, loading data through any of the standard load options (command line or tools) the data is automatically populated into the correct partitions.

Managing partitions

Once partitions are defined and created, you can manage your partitions with PROUTIL PARTITIONMANAGE. PARTITIONMANAGE has options that enable you to split, merge, truncate, rename, and alter the state of partitions. An additional option allows you to view the state or status of your partitions.

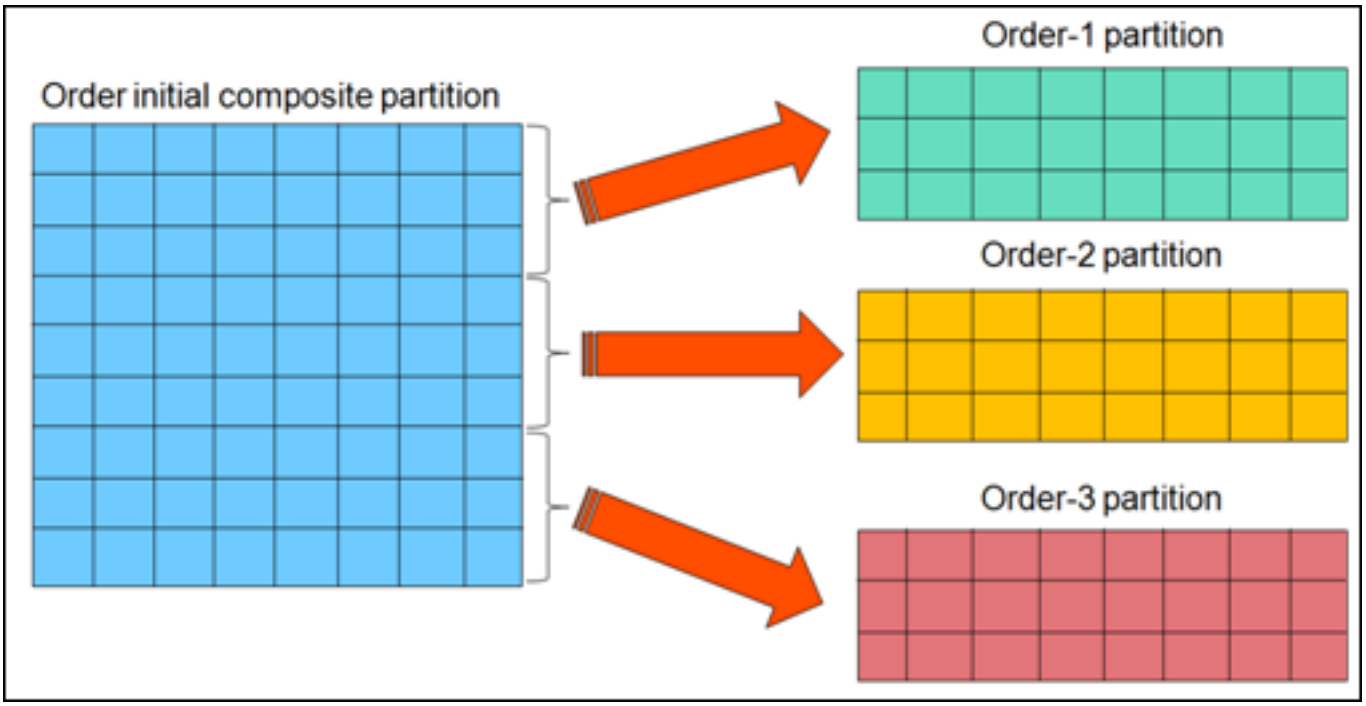
Splitting partitions

When you partition an existing table, the data is not automatically moved to the new partitions. PARTITIONMANAGE SPLIT is used to redistribute the records in one table partition into different partitions.

Use PARTITIONMANAGE SPLIT when a partition is too big, a partition needs to be partitioned into finer granularities, or for data migration when initially partitioning a table. When a table partition is split, the records, LOB fields, and local index partitions of the table partition instance are moved to the appropriate target partitions, and global indexes are updated.

The following diagram depicts the splitting a large table into smaller partitions.

Figure 51: Splitting partition example



PARTITIONMANAGE SPLIT relies on the target partitions being previously defined and designated as a split target. Designating a partition as a split target can be done through an OpenEdge SQL or ABL program, or the easiest way is through the OpenEdge Management/OpenEdge Explorer interface. (See *Management and OpenEdge Explorer: Managing Table Partitioning in Databases,* or the online help for these tools.) Once designated as a split target, existing data destined for the target is not accessible. For this reason, do not designate partitions as a split target until ready to execute the split.

PARTITIONMANAGE SPLIT allocates target partitions that are not allocated at the beginning of the split.

The syntax for PARTITIONMANAGE SPLIT is as follows:

```
proutil db-name -C partitionmanage split table table-name
  { partition table-partition-name | composite initial }
  [ useindex index-name ] [ recs numrecs ]
```

If you are splitting a table for the first time, source partition is specified as `composite initial`; for all other splits, the source partition is specified with `partition table-partition-name`. For complete details on PARTITIONMANAGE SPLIT, see the section, PROUTIL PARTITIONMANAGE SPLIT qualifier on page 752.

Merging partitions

After partitioning a table, you may find that your partition scheme needs to be consolidated. Reasons for consolidating partitions are that the range is too small, or perhaps that as data partitioned on a date range becomes more infrequently accessed, it is more efficient to manage larger partitions. You can use PARTITIONMANAGE MERGE to consolidate two or more range (or list-range) partitions into a single partition, provided the partitions are adjacent in the range definition.

The syntax for PARTITIONMANAGE MERGE is as follows:

```
proutil db-name -C partitionmanage merge table table-name
  partition table-partition-name partition table-partition-name [ partition
table-partition-name ]
 [ useindex index-name ] [ recs numrecs ]
```

PARTITIONMANAGE MERGE accepts two or three partitions as input. The utility treats the input as follows:

- If you specify two partitions, the first partition is the "start" partition, and the second is the "end" and implicit the "target" partition. All partitions between the start and end are merged into the end partition.
- If you specify three partitions, the first partition is the "start" partition, and the second is the "end", and the third partition is the "target" partition. All partitions between the start and end are merged into the target partition.

For complete details on PARTITIONMANAGE MERGE, see the section, PROUTIL PARTITIONMANAGE MERGE qualifier on page 750.

Truncating partitions

After a period of time managing a partitioned database, you may find that some of your partitions are no longer needed. For example, if the data in a partition has been archived, or perhaps all your data has been removed from the composite initial partition. You can use PARTITIONMANAGE TRUNCATE to reclaim the storage space of a partition, and optionally deallocate the partition.

The syntax for PARTITIONMANAGE TRUNCATE is as follows:

```
proutil db-name -C partitionmanage truncate table table-name
    { partition table-partition-name | composite initial }
    [ recs numrecs ] [ deallocate ]
```

PARTITIONMANAGE TRUNCATE removes any existing rows in the table partition, existing LOB data, and removes the key entries in the local indexes associated with the partition. If you specify deallocate, at the conclusion of truncating the partition, the data partition and local indexes are changed to the deallocated state.

For complete details on PARTITIONMANAGE TRUNCATE, see the section, PROUTIL PARTITIONMANAGE TRUNCATE qualifier on page 753.

Renaming partitions

After a period of time managing a partitioned database, particularly after a split or merge operation, you may find that the partition name is no longer representative of its contents. You can rename a partition with PARTITIONMANAGE RENAME.

The syntax for PARTITIONMANAGE RENAME is as follows:

```
proutil db-name -C partitionmanage rename partition table-partition-name
  partition new-table-partition-name
```

When you use PARTITIONMANAGE RENAME to rename a partition, the following rules apply:

- You can not rename the composite initial partition of a table.
- You can not rename a partition that is the source of merge operation.
- The *new-table-partition-name* must follow the standard rules for partition naming. The name:
  - Must be unique; it cannot already exist in the database
  - Must be 32 or fewer characters
  - Can only contain the following characters: a-z, A-Z, 0-9 (ASCII) , and the special characters::# $ % & - (dash) _ (underscore)
  - Is not blocked from using reserved words (as the database does for table names)

For complete details on PARTITIONMANAGE RENAME, see the section, PROUTIL PARTITIONMANAGE RENAME qualifier on page 751.

Altering partition state

As you manage a partitioned database, you may find that you may want to set a partition to a read-only state. Or you may want to change a read-only partition to a read-write state. You can use PARTITIONMANAGE ALTER to change the state of a partition.

A partition can be in one of two states:

- **read-only** — The **data** of the partition can not be changed. Language clients can only read the data. Limited utility access is permitted.
- **read-write** — Language clients can read, write, create, and delete data in the partition (provided that access is not restricted at a higher level, such as user-level security with a `can-*` permission in ABL).

The syntax for PARTITIONMANAGE ALTER is as follows:

```
proutil db-name -C partitionmanage alter table table-name
  { partition table-partition-name | composite initial }
        state { read-only | read-write }
```

Note: Before you use PARTITIONMANAGE ALTER you must enable it with PROUTIL ENABLEREADONLYPARTITIONS. See the section, PROUTIL ENABLEREADONLYPARTITIONS qualifier on page 703.

For complete details on PARTITIONMANAGE RENAME, including all the details on the allowed and restricted utility access to a read-only partition, see the section, PROUTIL PARTITIONMANAGE ALTER qualifier on page 746.

Viewing partitions

To aid managing a partitioned database, you can use the PARTITIONMANAGE VIEW on the command line to view the state and status of the partitions of your database. One example where PARTITIONMANAGE VIEW can be helpful is gauging the progress of a split or merge operation.

PARTITIONMANAGE VIEW reports on the following:

- The current state of a partition or partitions
- The progress of a partition utility operation (such as split, merge, and truncate)
- A list of a table's partition definitions, partition by partition in adjacent order

The syntax for PARTITIONMANAGE VIEW is as follows:

```
proutil db-name -C partitionmanage view
    [ table table-name
    [ partition table-partition-name | composite initial ] ]
    { list | state | status }
```

The options for PARTITIONMANAGE VIEW display different aspects of the queried table or partition, as follows:

- **`list`** — Lists partitions by partition definition order (adjacent definitions not by partition Id).
- **`state`** — Reports on the state of a partition. Partitions states are:
  - **No state reported (blank)** — The partition has no special characteristics, and is readable and writable.
  - **Rebuilding** — The partition is involved in a partition rebuild operation. The rebuilding type indicates the type of rebuild (split, merge, truncate, etc) and if the partition is the source or target.
  - **Read Only** — The partition is set to read only.
  - **Deallocated** — The partition is deallocated.
- **`status`** — Reports on the status of a partition that is in a rebuilding state. The number of rows in each partition is reported and can be used to measure progress of a partition operation such as split or merge.

For complete details on PARTITIONMANAGE VIEW, including sample output, see the section, PROUTIL PARTITIONMANAGE VIEW qualifier on page 755.

Copying partition data

The ability to copy partition data enables database administrators to recover accidentally deleted data from a partition without taking the entire database offline.

Once the need to restore data is discovered, the high-level restoration process is as follows:

1. Locally restore your most recent backup of your production database to a temporary location.
2. Roll forward the after-image files of your production database to your temporary database if your production database is enabled for after-imaging.
3. Move data from the temporary database to your production database.
4. Clean up.

The details of each step are discussed in the sections that follow.

Note: For a complete syntax description, see PROUTIL PARTITIONMANAGE COPY qualifier on page 748.

Restrictions on copying a partition

Restoring data by copying a partition has the following restrictions:

- Both the restored backup database and the target database must be enabled for table partitioning
- The restored backup database and target database must have identical schema for the partitions being copied
- Partitions being copied must be allocated in the backup database
- The tables must have indexes activated
- Dropped objects are not supported (only deleted data can be recovered, not table definitions)
- The restored backup database and the target database must be local (no remote connections)
- You can execute multiple partition copy commands simultaneously only if both databases are running in multi-user mode

Step 1: Restoring your database backup

The first step recovering deleted partition data is to restore your most recent backup of your production database to a temporary location.

When you restore your backup, you may want to protect the restored database from updates by other users. You achieve this by placing the database in a restricted access mode. You can enable restricted access to the database by adding the `-dbrestrict` parameter to the PROREST command. The syntax is:

```
prorest db-name . . . [ -dbrestrict restrict-type]
```

where *restrict-type* is one of the following values:

- **rollforward** — Indicating that the next allowable activity on this database is rolling forward after-image files
- **partitioncopy** — Indicating that the next allowable activity on this database is copying data with the PROUTIL PARTITIONMANAGE COPY utility

---

Note: All other `prorest` parameters are valid. Adding `-dbrestrict` to the `prorest` command enables the same restrictions as PROUTIL DBRESTRICT. See PROUTIL DBRESTRICT qualifier on page 675 for details of the restrictions.

The restriction you place on your database during the restore is determined by your use of after-imaging:

- If you use after-imaging, your next step in copying your data is applying after-image files to your restored database. Choose the value `rollforward` for *restrict-type*.
- If you don't use after-imaging, your next step in copying your data is executing the PROUTIL PARTITIONMANAGE COPY. Choose `partitioncopy` as the value for *restrict-type*.

Restricting access to the restored database ensures that no changes can be made to the database before the copy completes. If you don't restrict access, a user could connect and make changes, resulting in restoring incorrect data to your production database.

Step 1A: restricting access to your temporary database

If you don't restrict access to the database with the PROREST command, you can restrict access with the PROUTIL DBRESTRICT command as a separate step. The syntax is:

```
proutil db-name -C dbrestrict restrict-type restrict-action
```

For example, to restrict access for moving data on a database named `temporary_db`, you would enter:

```
proutil temporary_db -C dbrestrict datamove enable
```

For the complete syntax of PROUTIL DBRESTRICT, see PROUTIL DBRESTRICT qualifier on page 675.

Step 2: Rolling forward after-image files

The second step in copying a partition is to apply after-image files to your temporary database.

After restoring your production database backup to a temporary database, if you use after-imaging, you need to apply all the after-image files from the production database that have been created since the backup was created, up to the point just prior to when the accidental deletion occurred. If you don't use after-imaging, proceed to Step 3: Moving data with PROUTIL PARTITIONMANAGE COPY on page 383.

To expedite the process of rolling forward multiple after-image extents, RFUTIL ROLL FORWARD accepts a file containing a list of after-image files as input. The syntax of ROLL FORWARD with a list of extents is:

```
rfutil db-name -C roll forward -ailist listfile
```

The *listfile* file, containing a list of AI area files, must be structured as follows:

- Specify one AI file per line
- AI file names must have a fully specified absolute file name or a relative file name, relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

If a file in the list is not an after-image file, ROLL FORWARD issues a warning and the file is skipped.

The following example demonstrates applying the after image files from a database named `production_db` to a database named `temporary_db`.

To apply after-image files from a list:

1. Create a list of after-image files from `production_db`:

```
proenv>ls production_db.a* > production_ai_files.lst

proenv>cat production_ai_files.lst
production_db.a1
production_db.a10
production_db.a11
production_db.a12
production_db.a2
production_db.a3
production_db.a4
production_db.a5
production_db.a6
production_db.a7
production_db.a8
production_db.a9
```

   Notice that the files are sorted using an alphabetical sort. ROLL FORWARD sorts the files into the proper sequence order before applying them to the target database.

2. Apply the source after-image files to the target, `temporary_db`, as shown:

```
proenv>rfutil temporary_db -C roll forward -ailist
           production_ai_files.lst
```

   ROLL FORWARD sorts and applies the after-image files, reporting the progress of each file as it is applied to both the screen and the database log file.

Step 3: Moving data with PROUTIL PARTITIONMANAGE COPY

The third step in recovering deleted data is to copy data from your temporary database to your production database.

Once you have a temporary database restored from backup and rolled forward, you can use it as the source database for recovering your deleted data. You copy data into your production database with PROUTIL PARTITIONMANAGE COPY. You can issue a complete command from the command line, or you can invoke the menu-driven user interface.

A single command line copy allows you to copy data for one partition .

Depending on the data you need to copy, choose one or more of the following options to PROUTIL PARTITIONMANAGE COPY:

- Copy one partition with the PARTITIONMANAGE COPY command line on page 384
- Copy multiple partitions with the PARTITIONMANAGE COPY user interface on page 384

By default, PROUTIL PARTITIONMANAGE COPY, copies data by bundling 100 records in a single transaction. You can change the number of records per transaction on the command line with the `recs` parameter..

See PROUTIL PARTITIONMANAGE COPY qualifier on page 748 for the complete syntax.

Copy one partition with the PARTITIONMANAGE COPY command line

To copy one partition with the PARTITIONMANAGE COPY command line, the syntax is as follows:

```
proutil target-database -C partitionmanage copy source source-database
   { partition partition-name |
        composite initial table table-name }
     [ recs n]
```

For example, if you want to copy data to a production database named `production_db`, for the `customer_a_llr_1` partition , and your source database is named `temporary_db`, enter the following command:

```
proutil production_db -C partitionmanage copy source temporary_db partition
customer_a_llr_1
```

You can also copy a composite intial partition. For example, if you want to copy data to a production database named `production_db`, for the composite initial partition of the table `customer`, and your source database is named `temporary_db`, enter the following command:

```
proutil production_db -C partitionmanage copy source temporary_db composite initial
table customer
```

Copy multiple partitions with the PARTITIONMANAGE COPY user interface

You can invoke the PARTITIONMANAGE COPY user interface to make multiple selections of partitions to copy. Invoke the command line using the following syntax:

```
proutil target-database -C partitionmanage copy source source-database
       [ recs n]
```

For example, if your target database is named `production_db`, and your source database is named, `temporary_db`, start the user interface with the following command:

```
proutil production_db -C partitionmanage copy source temporary_db
      [ recs n]
```

The user interface menu appears as shown:

```
Progress OpenEdge Partition Copy Utility

Source Database     : temporary_db
Destination Database: production_db

1. Copy Table instance by Partition
2. Quit

Enter your selection:
```

From this menu, select option `1` to proceed or `2` to quit. The steps to proceed are as follows:

1. From the main menu, choose `1`.

The following screen appears:

```
   Choose Table Partitions for Partition Copy
-------------------------------------------------------------------------------------------
Option|  Table Name  |     Table Owner  |    Partition Name : ID          |Number
-------------------------------------------------------------------------------------------

     1|customer        |PUB               |cust_id_llr_1:1    |    1
     2|customer        |PUB               |cust_id_llr_2:2    |    1
     3|customer        |PUB               |cust_id_llr_3:3    |    1
     4|customer        |PUB               |cust_id_llr_4:4    |    1
     5|customer        |PUB               |cust_id_llr_5:5    |    1
     6|customer        |PUB               |cust_id_llr_6:6    |    1
     7|customer        |PUB               |cust_id_llr_7:7    |    1
     8|customer        |PUB               |cust_id_llr_8:8    |    1
     9|customer        |PUB               |cust_id_llr_9:9    |    1
    10|customer        |PUB               |cust_id_llr_10:10  |    1
    11|customer        |PUB               |cust_id_llr_11:11  |    1
    12|customer        |PUB               |cust_id_llr_12:12  |    1
    13|customer        |PUB               |cust_id_llr_13:13  |    1

-------------------------------------------------------------------------------------------
Page 1 of 31
                              Input Your Choice
 (c/C) - Change Display
 (a/A) - All   (n/N) - Next Page     (q/Q) - Quit   (g/G) - Go
```

This screen, and the following pages, display all table partitions, including composite partitions, in the database in alphabetical order.

2. Choose partitions by entering the number at the beginning of the row and pressing return. An asterisk will appear next to your selection.

3. Continue selecting partitions by number until you have selected all the partitions you want to copy.

Note: To undo a selection, enter the number a second time.

4. Once all the partitions to copy are selected, enter `g` or `G` to go.

Once you enter `g`, all the partitions that were selected are copied sequentially. Pressing `ctrl-C` cancels the process.

Step 4: Partition copy clean up

Once you have completed copying your partition(s) with PROUTIL PARTITIONMANAGE COPY, you can remove the access restriction from your temporary (source) database and delete it.

Remove access restriction from your source database

Before executing PARTITIONMANAGE COPY, you restricted access to the source database (temporary restored backup). Access continues to be restricted until you disable the restriction.

To remove the access restrictions from your source database:

1. If you started a server for source database, shut down the server.
2. Once the server is shut down, query the status of the datamove restriction with PROUTIL DBRESTRICT.

   For example, if the source database is named `temporary_db`, enter the following:

   ```
   proutil temporary_db -C dbrestrict partitioncopy status
   ```

   If the database is restricted, the following is displayed:

   ```
   Restricted Access Mode is enabled for PartitionCopy.
   ```

3. Remove the restriction with PROUTIL DBRESTRICT.

   Entering the following command:

   ```
   proutil temporary_db -C dbrestrict partitioncopy disable
   ```

   The results of removing the restriction is as follows:

   ```
   Database Restricted Access mode has been disabled for PartitionCopy
   ```

   You can now delete the temporary database if it is no longer needed.

Clear partition copy pending state

The `clear all` option for PARTITIONMANAGE COPY removes all partition copy pending state from the all partitioned objects that have this object state. The `clear all` option is necessary only in the rare cases that your database needs to be used with a version of OpenEdge earlier than Release 11.6.0 and a PARTITIONMANAGE COPY operation terminated prior to completion.

To remove any indicators of a partition copy pending state, and to disable the ability to continue to executing PARTITIONMANAGE COPY on the database, use the `clear all` parameter to PARTITIONMANAGE COPY. Invoke the command line using the following syntax:

Syntax

```
proutil target-database -C partitionmanage copy clear all
```

For example, to clear a database named `production_db`, enter the following:

```
proutil production_db -C partitionmanage copy clear all
```

The results of executing a `clear all` are similar to the following:

```
Partition Copy is attempting to clear "in process partition copy" object state from all
 partitioned objects:(17927)
Index 1 of table 1 partition 1 in area 22 is no longer in the process of being copied
by Partition Copy.(17936)
Lob 3 of table 1 partition 1 in area 22 is no longer in the process of being copied by
 Partition Copy. (17936)
Partition 1 of table 1 in area 22 is no longer in the process of being copied by
partition copy.(17937)
Partition Copy has successfully cleared "in process partition copy" object state and
has disabled the feature.(17930)
```

Partition-level database administration

Table partitioning enables a database administrator to perform some maintenance tasks at one partition at a time. The advantage of most partition-level maintenance is that it can be performed online, and when executed online, the other partitions of a table not being operated on remain accessible by the language clients and other utilities.

Partition-scoped utilities

Existing database utilities that are enhanced to operate on a partition include the following:

- Data utilities
  - BULKLOAD
  - DUMP
  - DUMPSPECIFIED
  - LOAD
  - TABLEMOVE
- Index utilities
  - IDXACTIVATE
  - IDXBUILD (offline)

  Note: TPIDXBUILD rebuilds partition indexes when the database is online.

  - IDXCHECK
  - IDXCOMPACT
  - IDXDEACTIVATE
  - IDXFIX
- Create and toss limits
  - SETLOBCREATELIMIT
  - SETLOBTOSSLIMIT
  - SETTABLECREATELIMIT
  - SETTABLETOSSLIMIT

See the complete description of each utility in PROUTIL Utility on page 629 for details on how to specify a partition with each utility.

Table-partitioned index rebuild

For databases enabled for table partitioning, the TPIDXBUILD utility allows database administrators to rebuild an index or indexes for a partition while the database remains online, in addition to offline. TPIDXBUILD only rebuilds indexes for a table partition.

Table-partitioned index rebuild acquires an exclusive table partition lock when performing the rebuild, blocking other processes from creating or deleting index objects. Once the rebuild starts, clients cannot use the index in a query, locked or no-lock.

To prevent users from accessing a rebuilt index with a stale cursor, ABL clients with a timestamp older than a current timestamp from TPIDXBUILD, must log out.

Table-partitioned index rebuild has significant differences from the full index rebuild (PROUTIL IDXBUILD). Those differences include:

- Table-partitioned index rebuild can operate while the database is online.
- You can still access the database if the index rebuild fails. If the index rebuild fails, the index is left offline and must be brought online with an index utility (index rebuild or activation) because not all of the index rebuild activity is logged. Users can access the database after a failed table-partitioned index rebuild, provided the new process goes through crash recovery.
- You cannot use table-partitioned index rebuild to clear a tainted index flag. Only the full index rebuild can do that.
- Table-partitioned index rebuild is not multi-threaded.
- To improve performance, table-partitioned index rebuild deletes and rebuilds the index.
- There is no menu-driven option to TPIDXBUILD, you must completely specify the rebuild on the command line.

You can track the progress of TPIDXBUILD using the User status (`_UserStatus`) Virtual System Table.

18

Change Data Capture

OpenEdge Change Data Capture is an OpenEdge RDBMS feature that identifies and captures data that has changed in tables of a source database, as a result of create, update, and delete (CUD) operations. Generally speaking, Change Data Capture is an industry term that describes the process of duplicating subsets of OLTP data in an external data source with a relatively up to date version of relational data. The OpenEdge implementation of CDC provides a flexible and scalable capture process to facilitate the data extraction, transformation, and eventually the loading of the data to an external data source. This is useful to customers who need to export data to a data warehouse or business intelligence application. The data provided by the capture process exists on the source database and is maintained in relational form.

The details of CDC are described in *OpenEdge Getting Started: OpenEdge Change Data Capture Guide*. In the sections that follow, database-specific topics are discussed.

For details, see the following topics:

- Change Data Capture highlights
- CDC and other RDBMS features
- Enabling CDC
- Using the CDC cache

Change Data Capture highlights

The strengths of OpenEdge CDC, include its flexibility and scalability.

OpenEdge CDC is flexible because:

- Captured data is maintained in the same database
- Captured data is maintained in relational form
- Captured data is accessible with SQL and ABL
- No changes are required in your application

OpenEdge CDC is scalable because you can define CDC policies such that:

- The amount of data captured is variable by table
- You can capture no data, some data, or the whole record
- You can index the data for easier retrieval

The amount of data captured is controlled through policies defined at the table and field level.

Change Data Capture is a licensed product. It requires either a combination of an RDBMS license (Workgroup or Enterprise) and the corresponding CDC license, or the Advanced Enterprise RDBMS license.

CDC Policies

Defining the data captured by CDC is done through policies. CDC policies are created to define the tables and fields to be tracked. Create policies in OpenEdge Management or OpenEdge Explorer. For details on creating policies in OpenEdge Management, see *OpenEdge Management and OpenEdge Explorer: Managing Change Data Capture in databases*.

CDC tables in your database

CDC tracks and records the data and changes in tables within your database.

- **Schema tables**

  There are two metadata tables: _Cdc-Table-Policy and _Cdc-Field-Policy. These tables are created in the source database when you enable Change Data Capture for the database. The metadata tables contain information about CDC policies for tables and their associated fields. You do not have to allocate space for the metadata tables; these tables reside in the schema storage area of the database. You should not manage these tables, they are managed by OpenEdge RDBMS.

  The relationship between the CDC Table Policy Table and the Field Policy Table is a 1- to-many relationship. The hierarchy consists of two levels: Table policy (_Cdc-Table-Policy) and Field policy (_Cdc-Field-Policy). The table policy is a parent and the field policies are its children.

  The CDC Table policy describes the policy attributes such as filename of the source table reporting to this policy, area the table resides in, active status of policy, and coverage level of the policy. This policy can point to a number of CDC Field policy records that identify the fields to be captured in the change table for this table. Once the new policy is defined, created, and committed, the policy becomes active. Each time a CUD record operation is executed, a record is written to the Change Tracking Table and the Change Table assigned to the policy.

- **Data tables**

  There are CDC data tables: the _Cdc-Change-Tracking table, and an individual CDC change table for each source table for which you want to capture change data. Database administrators are responsible for allocating space for all CDC data tables, and specifiying the area when enabling CDC.

  A CDC change table is created in the source database when you define and activate a CDC policy for a source table. Note that a CDC change table is created only when the policy level is higher than minimal. CDC policies with the minimal level do not have a change table; their data is written only to the _Cdc-Change-Tracking table.

The _Cdc-Change-Tracking table maintains a record of all change capture activities for *all* tables with CDC policies. This table is created in the source database when you enable Change Data Capture for the database.

For details on these tables, their uses, and their schema, see *OpenEdge Getting Started: OpenEdge Change Data Capture Guide*.

CDC and other RDBMS features

In the sections that follow, the interactions of CDC with RDBMS features and utilities is described.

RDMBS Features

The interaction of CDC with other database features is described in the table that follows.

Feature	Behavior
Auditing	Both Auditing and CDC can be configured for the same data tables, however: • Auditing of CDC data tables is not supported. • CDC of audit policy tables is not supported.
Replication	CDC can be enabled and configured on a Replication target database, however, the Replication source database(s) must have the areas that CDC uses, or the replication of the CDC enablement will fail. If the database structures match, Replication sends all CDC updates to the target database(s). Because Replication targets are read only, no CDC activity originates on the target. When the source database is enabled at the feature level and the Change Tracking Table is created by the enable utility, notes are written to the recovery logs. The notes are communicated to the target database via the replication process and the target is then enabled for CDC feature. When tables are enabled for change capture and their change tables are created on the source database, recovery notes are logged and communicated to the replication target. The notes recreate the change tables for each CDC table on the target.
Encryption	OpenEdge TDE Encryption can occur on the area or table level. If encryption is desired for a CDC change table, CDC change tables must be created in an encrypted area if the source data table is already encrypted. Encryption must also allow CDC change tables to be encrypted to the same table level of encryption as the source data table.
Multi-tenancy	CDC and Multi-tenancy coexist in a database without issue.
Table partitioning	CDC and Table Partitioning coexist in a database without issue.
Security	Security roles for CDC are not necessary and are not provided. However, if the Change Tracking table and Change table(s) need additional security, you can add CAN constraints to these tables.

International databases

The Change Tables are writeen in the code page of the database, so as long as any extraction process retrieves the data with a related client code page there will be no issue with support. It is expected that the extraction application will use the same code page as used when accessing the source table.

Database utilities not impacted by CDC triggers

The interaction of CDC with utilities that are known to change rowid, tenant Id, and partition Id is described in the table that follows. These utilities do not change the data they manage, however certain utility operations will generate Change Tracking and Change Records by the nature of their changes.

Utility	Utility can run on CDC tables	Behavior when the Source Table is Enabled for CDC	Change tracking data generation[6]
Tablemove	Yes	Movement of records from one area or partition to another does not write Change Table records. In addition, rowid changes when the record is moved to the new location. No special Change record is recorded for this change.	No CTR, No CR
Datamove	No	Movement of records from one database to another do not write Change Table records.	No CTR, No CR
Partition Copy	No	Movement of records of one partition and database to another do not write Change Table records.	No CTR, No CR
Schema mover	No	Changes the area of source data records that previously resided in the schema area. It does not move source data records, so no Change Table records are written.	No CTR, No CR
Backup/Restore	No	Utility operates at Block level, so no CDC activity is necessary.	No CTR, No CR
AuditReconfig	No	Schema records, such as audit schema, cannot be enabled for CDC, so no Change data is generated by this utility.	No CTR, No CR

[6] **For values in this column, CTR indicates Change Tracking Record, and CR indicates Change Table.**

Utility	Utility can run on CDC tables	Behavior when the Source Table is Enabled for CDC	Change tracking data generation[6]
Partition Split	No	Partition Split can change partition Id, but does not generate a Change record to record the event.	No CTR, No CR
Partition Merge	No	Partition Merge causes a drop partition. there is a change tracking record created.	CTR
Partition Rename	No	The partition name changes and not generate a Change Tracking or a Change Tracking record. Changing a partition name does not affect the partition Id or the record Id.	No CTR, No CR
Table Rename, Field Rename (Language Only)	No	The languages' changing of the Table name or a field name does not generate a Change Tracking record or Change Table record.	No CTR, No CR
Binary Load, Bulkload	No	Could cause rowid, partition id changes and does generate a Change Tracking and Change Record. **Note:** Loading Change Table schema and data does not produce a CDC enabled Change Table.	CTR, CR

[6] **For values in this column, CTR indicates Change Tracking Record, and CR indicates Change Table.**

Utility	Utility can run on CDC tables	Behavior when the Source Table is Enabled for CDC	Change tracking data generation[6]
Dictionary Load, SQL Load	Yes	Could cause rowid, partition id changes and generates a Change Tracking and Change Record. **Note:** Loading Change Table schema and data does not produce a CDC enabled Change Table.	CTR, CR
Binary Dump	No	Could cause rowid, partition id changes but does not generate a Change record.	No CTR, No CR
Data Dictionary Dump,SQL Dump	Yes	Could cause rowid, partition id changes but does not generate a Change record.	No CTR, No CR
Area Truncate	No	Deallocated records do not generate Change records. The Change Tracking Table is not allowed to participate in an area truncate operation because it contains records for all tables participating in CDC.	No CTR, No CR
Convchar	No	The data is not changing, its encoding is changing, but it is still the same data, so there is no change to reflect.	No CTR, No CR

Database utilities impacted by CDC triggers

The interaction of CDC with utilities that are known to change data in CDC source tables is described in the table that follows.

Utility	Behavior when the Source Table is Enabled for CDC	Change tracking data generation[7]
Idxfix Option 6 delete recid	Record deletion is captured.	CTR, CR

6 **For values in this column, CTR indicates Change Tracking Record, and CR indicates Change Table.**

7 **For values in this column, CTR indicates Change Tracking Record, and CR indicates Change Table.**

Utility	Behavior when the Source Table is Enabled for CDC	Change tracking data generation[7]
MT Table Deallocate	A de-allocate Change Tracking record is generated. This operation occurs for a tenant and records are de-allocated as a storage object for the tenant. Multiple delete records are not called for this operation.	CTR
Partition Truncate	A de-allocate Change Tracking record is generated. This operation occurs at the partition level and records are de-allocated as a storage object for the partition. Multiple delete records are not called for this operation.	CTR
Tenant Delete/Drop	A drop tenant Change Tracking record is generated. MT tenant drop is a ABL/SQL operation. The operation can be called on allocated/de-allocated partitions. Group partitions also generate a Change Tracking Record when they deleted.	CTR
Partition Drop (Language action only)	A partition drop causes a Change Tracking record to be generated when the languages execute this action. This also works for a composite partition.	CTR

Database utilities that process CDC Tables

The utilities listed in the table that follows can execute on CDC schema for maintenance or reporting. None of the utilities generate Change Tracking or Change Table records.

Utility	Behavior when the Source Table is Enabled for CDC
Idxbuild, idxcheck	Can rebuild indexes for Change Schema, Change Table
Idxfix	Can fix indexes for Change Schema, Change Table
Idxactivate	Can active the Change Tracking Table indexes. Indexes in the Change Table cannot be deactivated by a DBA.
Dbtool	Record format fixups – physical changes allowed

[7] **For values in this column, CTR indicates Change Tracking Record, and CR indicates Change Table.**

Utility	Behavior when the Source Table is Enabled for CDC
Dbanalys (all analysis types)	Includes all CDC related tables
Tablemove	Can move the Change Table only

Enabling CDC

Before you can use CDC, you must enable it, and before you enable it, you should allocate space for the associated tables and indexes. The _Cdc-Change-Tracking table is the repository of all change capture data in a database. Once Change Data Capture is running, _Cdc-Change-Tracking can grow very quickly and become very large. It is important that you allocate enough space for this table and its indexes.

To allocate space for CDC in your database, add new areas with PROSTRCT ADD. You can use a structure file similar to the following:

```
#
# add_cdc.st
#
d "CDC_Track_Data":20,64;512 . f 102400
d "CDC_Track_Data":20,64;512 . f 102400
d "CDC_Track_Data":20,64;512 .
#
d "CDC_Track_Idx":21,1;64 . f 10240
d "CDC_Track_Idx":21,1;64 . f 10240
d "CDC_Track_Idx":21,1;64 .
#
```

And add it to your database, for example:

```
prostrct add db-needing-cdc add_cdc.st
```

After adding the areas for the CDC data and indexes, enable your database as follows:

```
proutil db-needing-cdc -C enablecdc
     area CDC_Track_Data indexarea CDC_Track_Idx
```

In order to enable or disable the feature, the Table and Field Policies must be removed in a separate operation beforehand. Removing Table and Field policies causes all related Change Tables to be removed. Removing policies can be accomplished through OpenEdge Management or OpenEdge Explorer. The Policies and their related Change Tables can also be removed by a language table drop operation on the Source Table. The Feature Bit mechanism activated at enablement restricts direct connections to a CDC-enabled database from older releases.

For details on using the ENABLECDC command, see PROUTIL ENABLECDC qualifier on page 698

Using the CDC cache

Performance of Change Data Capture (CDC) can be impacted by the size of the CDC cache.

To capture data changes in CUD operations, CDC triggers are embedded in the operations to inspect changed data and record them. The changes are processed according to user-defined CDC policies to determine if the change needs to be recorded and the level of detail. As CDC policies and their related meta-schema are frequently used information and not often changed, most of this information is stored in a shared cache, while some information is stored in users' private cache to improve performance.

CDC cache is used by CDC triggers to improve performance when capturing data changes and writing captured data. While cached information and maintenance operations are internal, the CDC cache requires reserved shared memory to store the information, and is customizable. Users can use `-cdcsize n` to reserve memory for the CDC cache. A large cache size reduces the chance of eviction and paging that decrease performance. Allocate enough space to improve performance. For details on the `-cdcsize` parameter, see CDC cache size (-cdcsize) on page 491.

The CDC cache allocation is split between a primary and secondary cache. The primary cache is read-only, and loaded with the CDC policies that exist at startup. All updates are performed in the secondary cache. The size of the primary cache is fixed at startup, and in most cases is set to 75 percent of the specified size. If the remaining 25 percent is not enough space for the secondary cache, you can use PROUTIL INCREASETO to increase the size of the secondary cache online.

In CDC, each change needs to be captured to assure data integrity. The CDC cache aids the performance of CDC triggers. The CDC cache has the following attributes:

- Cached information is stored in shared memory.
- CDC-enabled tables have information stored in the cache whether the policies on the table is active and current or not.
- CDC triggers use the cache in forward processing for CUD operations.
- The CDC cache manager assures cached information is synchronized with policies changes.
- If eviction is necessary, out of date entries in the cache are evicted and replaced with new information when they are refreshed by a caller.
- For replacement, if the cache does not have room for a new entry, the Least Recently Used entry (LRU) is replaced with the new one.

19

Diagnostic Data Collection

Diagnostic Data Collection allows for a variety of critical information to be saved when certain events occur. Collecting this data can aid in the analysis of the root cause of an event. Almost all aspects of this feature are customizable, including: the events for data collection, the amount of data collected, and the formatting of the data.

For details, see the following topics:

- Events
- Data Files
- Protrace data
- DiagProc file
- Diagnostic data startup parameters
- Diagnostic data collection in PROMON
- Diagnostic data output file formats

Events

Diagnostic Data Collection occurs when a triggering event is encountered. A *triggering event*, for the purposes of Diagnostic Data Collection, is one of the following:

- Lock table overflow
- BI growth
- System errors and/or crashes

Each event records a unique entry in a diagnostic event tracking log file before recording the requested data. This unique entry can be used to match the event id associated with each event, with the output directory containing any additional requested diagnostic data output.

For any occurrence of a triggering event, there will be consistent storage and formatting of the data collected. The amount of data collected depends on the event level. The formats of the output files are described in Diagnostic data output file formats on page 407. The triggering events also provide an alert mechanism and the option to invoke an external executable. The executable can provide the DBA the opportunity to perform additional data collection or processing. Lastly, a configurable pause, provides time to the DBA to optionally perform additional manual actions prior to the event being processed.

Specifying an event

Triggering events can be specified in server startup parameters, and can be modified at runtime with PROMON or by editing the _dbParams VST.

To specify the triggering event at startup, use the `-diagEvent` parameter. The `-diagEvent` accepts a character string of comma-separated keywords and a data collection level that identify the triggering event(s) to capture data for and the amount of data.

The accepted keywords for `-diagEvent`, are:

- `locktable`
- `bithold`
- `syserror`

You specify the level of data collection by adding `:n` to the event. Valid values for `n` range from -15 to 15.

The following example demonstrates specifying the locktable and syserror triggering events:

```
-diagEvent locktable:3,syserror:-15
```

If you do not specify an event, it is set with the value specified by `-diagEvtLevel`. By default, `-diagEvtLevel` is 0, meaning that data collection for that event is off.

Event Levels

For every triggering event, you can specify a level. The level dictates the amount of data collected when the even occurs.

The following table describes the level settings.

Event Level (+/-)	Data collected	Spawn process
0	None	No
1	Summary data	No
2	Summary Data Detailed Data	No
3	Summary Data Detailed Data Protrace collection	No
4	None	Yes
5	Summary data (1+4)	Yes
6	Summary data Detailed Data (2+4)	Yes
7	Summary data Detailed Data Protrace collection (3+4)	Yes
8	Record data of all other enabled events	No
9	Summary Data Record data of all other enabled events (1+8)	No
10	Summary Data Detailed Data Record data of all other enabled events (2+8)	No

Event Level (+/-)	Data collected	Spawn process
11	Summary Data Detailed Data Protrace collection Record data of all other enabled events (3+8)	No
12	Record data of all other enabled events	Yes
13	Summary data Record data of all other enabled events (1+12)	Yes
14	Summary data Detailed Data Record data of all other enabled events (2+12)	Yes
15	Summary data Detailed Data Protrace collection Record data of all other enabled events (3+12)	Yes

Note:

The protrace collection is only available on UNIX.

Negative values produce the same output and process spawning option as the positive value, but also indicate that the event should only trigger if the error is fatal, bringing the database down.

Data Files

The sections that follow describe the files created by diagnostic data collection.

Tracking log file

The first file created is the diagnostic event tracking log file. By default, this file is created in the database directory. You can change the location of the tracking log file using one of the following methods:

- Specify the directory at startup with `-diagDir`.
- Change the value while the database is online with PROMON.
- Change the value while the database is online by updating the value of the `_dbparams` VST field for the `-diagDir` parameter.

By default, the event tracking file is named, `diagEvent_Tracking.csv`. If you specify JSON as your output format, the file is named `diagEvent_Tracking.json`. You can change the file prefix (`diagEvent_`) using one of the following methods:

- Specify the prefix at startup with `-diagPrefix`.
- Change the value while the database is online with PROMON.
- Change the value while the database is online by updating the value of the `_dbparams` VST field for the `-diagPrefix` parameter.

Event sub-directory

When a triggering event occurs, and the level is non-zero, a sub-directory is created to hold the event-specific files. The sub-directory name is constructed as follows:

```
diagEvent_+ YYYY-MM-DDThh:mm:ss.sTZDTZD_ + eventid
```

The *eventid* is the combination of the event and an identifier. The event is the name of a triggering event, or `_DbStatus` to indicate that the data is being collected by a directive, and not an event. The identifier is an increasing number, that starts from 1, each time the database is brought online.

Lock table event files

For the lock table overflow triggering event, the following files are created for CSV output format, based on the level specified. For JSON output, only one file is created.

- Summary (level 1)

Filename	Contents
prefix`locktable_summary.csv`	Pid of all users with lock table entries, and a summary of locks held (_usrlock VST data)

- Summary and detail (level 2)

Filename	Contents
prefix`locktable_summary.csv`	Pid of all users with lock table entries, and a summary of locks held (_usrlock VST data)
prefix`locktable_detail.csv`	Contents of the _lock VST by user
prefix`transaction_detail.csv`	Contents of the _trans VST by user

BI threshold event files

For the BI threshold triggering event, the following files are created for CSV output format, based on the level specified. For JSON output, only one file is created.

- Summary (level 1)

Filename	Contents
*prefix*BILogging_summary.csv	• The size of each BI file • Cluster size and BI block size • Oldest transaction id • All Pids with an active transaction

- Summary and detail (level 2)

Filename	Contents
*prefix*BILogging_summary.csv	• The size of each BI file • Cluster size and BI block size • Oldest transaction id • All Pids with an active transaction
*prefix*BILogging_detail.csv	Contents of the _logging VST
*prefix*transaction_detail.csv	Contents of the _trans VST by user

Syserror event files

For the syserror triggering event, the following file is created for CSV output format. The files are the same, regardless of the level. For JSON output, only one file is created.

Filename	Contents
*prefix*locktable_summary.csv[8]	Pid of all users with lock table entries, and a summary of locks held (_usrlock VST data)

DiagComplete file

The DiagComplete file is not a data file, but a processing indicator. When a triggering event occurs, you can specify an amount of time to pause after collecting the data and processing the event. If you want to cut the pause time short, create a DiagComplete file in the event sub-directory. When the presence of this file is detected, the pause is ended, regardless of remaining time.

[8] The file contents are the same as those provided during lock table overflow, so the same file name is used.

Protrace data

For all triggering events, event level 3, or 3 in combination with another value, indicates collecting protrace data. For every local user, a protrace file is written to the user's working directory. The first time a protrace file is written for a user, a message is written to the database log file indicating the location of the protrace file.

DiagProc file

For all triggering events, event level 4, or 4 in combination with another value, a process is spawned after any diagnostic data files are written, and before the triggering event is processed. The spawned process executes the diagProc file. The diagProc file is an executable file that can accept three input parameters: the current event diagnostic directory name, the triggering event, and the database name.

On Unix, the diagProc file must be an executable or executable shell script named `diagProc`, and must reside in the database directory.

In Windows, the diagProc file must be an executable named `diagProc.exe`, or a batch file named `diagProc.bat`, and must reside in the database directory. The `diagProc.exe` file is searched for first, and `diagProc.bat` second. The first file found is executed.

Caution: An OpenEdge server calling an external executable or script is a potential security risk. Restricted access to the script is advised.

Diagnostic data startup parameters

You can tailor your diagnostic data collection from startup, with the parameters defined in the table that follows.

Table 51: Diagnostic data collection startup parameters

Parameter	Description	For more information, see...
Diagnostic directory (`-diagDir`)	Specify the directory where the diagnostic tracking log file is written and subdirectories for each diagnostic event are created.	Diagnostic directory (-diagDir)
Diagnostic events value (`-diagEvent`)	Specify the database events to collect diagnostic data for.	Diagnostic events value (-diagEvent)
Diagnostic event level (`-diagEvtLevel`)	Specify the "level" of diagnostic data to collect.	Diagnostic event level (-diagEvtLevel)
Diagnostic data format (`-diagFormat`)	Specify the format of the diagnostic data files.	Diagnostic data format (-diagFormat)
Diagnostic field separator (`-diagFS`)	Specify the separator to use when writing diagnostic data to a CSV-formatted file.	Diagnostic field separator (-diagFS)

Parameter	Description	For more information, see...
Diagnostic pause length (`-diagPause`)	Specify the number of seconds to pause after recording data for a diagnostic event before processing the event.	Diagnostic pause length (-diagPause)
Diagnostic prefix value (`-diagPrefix`)	Specify a common prefix to standard diagnostic data files.	Diagnostic prefix value (-diagPrefix)

Diagnostic data collection in PROMON

You can tailor your diagnostic data collection from within PROMON.

Diagnostic Data Collection menu

Access the Diagnostic Data Collection menu in PROMON as follows:

1. Start PROMON
2. Select **Advanced Options** by entering `R&D`.
3. Select **Administrative Functions** by entering `4`.
4. Select **Diagnostic Data Collection** by entering `14`.

The following menu appears:

```
05/16/17        OpenEdge Release 11 Monitor (R&D)
12:02:06        Diagnostic Data Collection

        Current Diagnostic Data Collection Settings:

         1. Diagnostic directory:       ""
         2. Diagnostic field separator: ' '
         3. Diagnostic pause time:      0
         4. Diagnostic prefix value:    diagEvent_
         5. Diagnostic report format    csv

         6. Lock Table Overflow:       0: Data collection disabled
         7. BI Threshold:              0: Data collection disabled
         8. System Error:              0: Data collection disabled

         9. Collect enabled diagnostic data now


Enter a number, P, T, or X (? for help):
```

For items **1-5**, change any value by entering the number, and entering a new value as prompted.

For items **6-8**, the following menu prompts you to set a value for the data collection.

```
Enter 0: Disabled
      1: Summary
      2: Summary & Detailed
      3: Summary, Detailed & Protraces
    4-7: Proc invocation in combination w/chosen option 0 thru 3
   8-15: Report all enabled event collectors in combination w/chosen option 0 thru 4
 (-15 to 15):
```

Note that negative values collect the same amount of data, but only in the case of a fatal error that will bring down the database.

For item **9**, the data collection executes immediately, and does not respect the pause time value.

Diagnostic data output file formats

The level of output requested determines the files created when an event occurs.

Diagnostic tracking file

A top-level diagnostic tracking file is created when the first event is triggered, and appended for every subsequent event. This file allows you to correlate an event sub-directory with a specific event.

If your output format is CSV, the diagnostic tracking file is formatted as follows:

```
"Timestamp" "PID" "TID" "Level" "ProcType" "Event" "Dbname" "EventDir" "EventId" "Status"
```

The `"Status"` element contains `START` or `END` to indicate the duration of the event.

If your output format is JSON, the diagnostic tracking file is formatted as shown in the following example:

```
{"AnEvent_2_Start":{"Timestamp":"2017-05-16T17:41:06.000-4:00","PID":"P-79844",
 "TID":"T-140067960612608","Level":0,"ProcType":"MON ","Event":"DbStatus",
 "Dbname":"/scratch/doc/117/docdb",
 "EventDir":"/scratch/doc/117/diagEvent_2017-05-16T17:41:06.000-4:00_DbStatus_2",
 "EventId":2,"Status":"Start"}}

{"AnEvent_2_End":{"Timestamp":"2017-05-16T17:41:06.000-4:00","PID":"P-79844",
 "TID":"T-140067960612608","Level":0,"ProcType":"MON ","Event":"DbStatus",
 "Dbname":"/scratch/doc/117/docdb",
 "EventDir":"/scratch/doc/117/diagEvent_2017-05-16T17:41:06.000-4:00_DbStatus_2",
 "EventId":2,"Status":"End"}}
```

Diagnostic BI threshold files

When an BI threshold event is triggered, depending on the event level, the following BI logging files are created in the event subdirectory.

If your output format is CSV, the BI logging summary file is formatted as follows:

```
"Timestamp" "EventId" "DbName" "BIIO" "CrashProt" "CommitDelay" "ClusterAge" "Blksize"
 "Clustersize" "NumExtents" "BISize" "BytesFree" "LastChkpt" "BIBufs" "FullBuffers"
"CurrCluster" "ClusterHWM"
```

If your output format is CSV, the BI logging detail file is formatted as follows:

```
"Timestamp" "EventId" "DbName" "TotalWrts" "BIWWrites" "RecWriten" "BytesWrtn" "TotReads"
 "RecRead" "BytesRead" "ClstrClose" "BBuffWaits" "EBuffWaits" "ForceWaits" "ForceWrts"
 "PartialWrts" "Trans" "UpTime" "InputBufHit" "OutputBufHit" "ModBufHit" "BOBufHit"
```

If your output format is CSV, the transaction detail file is formatted as follows:

```
"Timestamp" "EventId" "DbName" "UserNum" "TenantID" "TransID" "RLCounter" "State" "Flags"
 "JState" "JFlags" "XID" "Txtime" "Coord" "CoordTx" "Duration" "BIRecReads" "BIRecWrites"
```

If your output format is JSON, there is only one file in the event directory. The following example shows BI entries from a JSON-formatted diagnostic file:

```
{"BILogging_summary":{"BIIO":"Reliable","CrashProt":1,"CommitDelay":3,"ClusterAge":0,
 "Blksize":8192,"Clustersize":524288,"NumExtents":1,"BISize":3193,"BytesFree":482471,
 "LastChkpt":"2016-08-15T14:55:11.000-4:00]","BIBufs":20,"FullBuffers":5,"CurrCluster":2,

 "ClusterHWM":5},

"file_detail":{
"file_entry":{"FileName":"/usr1/richb/11/x.db","Size":640,"InUse":5,"Extend":512,"Blksize":8192,


"IOMode":"BOTHIO","Reads":5,"Writes":0,"Extends":0,"BufWrites":1,"UnbufWrites":0,"BufReads":7,

 "UnbufReads":0},
"file_entry":{"FileName":"/usr1/richb/11/x.b1","Size":3200,"InUse":400,"Extend":512,"Blksize":8192,


"IOMode":"UNBUFIO","Reads":15685,"Writes":67901,"Extends":1,"BufWrites":0,"UnbufWrites":67934,

 "BufReads":0,"UnbufReads":15686},
"file_entry":{"FileName":"/usr1/richb/11/x.d1","Size":2176,"InUse":256,"Extend":512,"Blksize":8192,

 "IOMode":"BOTHIO","Reads":67,"Writes":1018,"Extends":0,"BufWrites":1,"UnbufWrites":1018,

 "BufReads":69,"UnbufReads":0},
"file_entry":{"FileName":"/usr1/richb/11/x_7.d1","Size":128,"InUse":15,"Extend":512,"Blksize":8192,


"IOMode":"BOTHIO","Reads":1,"Writes":0,"Extends":0,"BufWrites":1,"UnbufWrites":0,"BufReads":3,

 "UnbufReads":0},
"file_entry":{"FileName":"/usr1/richb/11/x_8.d1","Size":4096,"InUse":511,"Extend":512,"Blksize":8192,


"IOMode":"BOTHIO","Reads":193,"Writes":30836,"Extends":0,"BufWrites":30837,"UnbufWrites":0,

 "BufReads":195,"UnbufReads":0},
```

Lock table overflow files

When a lock table overflow event is triggered, depending on the event level, the following output files are created in the event subdirectory.

If your output format is CSV, the lock table summary file is formatted as follows:

```
"Timestamp" "EventId" "DbName" "PID" "User Num" "Tenant" "Domain ID" "Curr Locks" "Lock
 HWM" "User Name" "TTY Device" "Program file" "Line Number"
```

If your output format is CSV, the lock table detail file is formatted as follows:

```
"Timestamp" "EventId" "DbName" "Lock Type" "Table #" "Rowid" "Partition #" "Hash Chain"
 "Flags" "PID" "User Num" "Tenant" "User Name" "Domain ID" "TTY Device" "Trans State"
"Trans Flags" "Trans ID"
```

If your output format is CSV, the transaction detail file is formatted as follows:

```
"Timestamp" "EventId" "DbName" "User Num" "Tenant ID" "Trans #" "RL Counter" "State"
"Flags" "JState" "JFlags" "XID" "Txtime" "Coord" "CoordTx" "Duration" "BIRecReads"
"BIRecWrites"
```

If your output format is JSON, there is only one file in the event directory. The following example shows lock table entries from a JSON-formatted diagnostic file:

```
"locktable_summary":{
"userlock_entry":{"UserNum":0,"UPID":10732,"Tenant":-32767,"DomainID":0,"UserName":"richb",

 "TTYDevice":"/dev/pts/5","CurrLocks":0,"LockHWM":1,"ProgName":"","Line":0},
"userlock_entry":{"UserNum":5,"UPID":7459,"Tenant":-32767,"DomainID":0,"UserName":"richb",

 "TTYDevice":"/dev/pts/7","CurrLocks":0,"LockHWM":1,"ProgName":"","Line":0},
"userlock_entry":{"UserNum":6,"UPID":7492,"Tenant":0,"DomainID":-4,"UserName":"richb",

 "TTYDevice":"/dev/pts/12","CurrLocks":3,"LockHWM":1107,"ProgName":"","Line":0},
"userlock_entry":{"UserNum":7,"UPID":7496,"Tenant":0,"DomainID":-4,"UserName":"richb",

 "TTYDevice":"/dev/pts/13","CurrLocks":0,"LockHWM":1126,"ProgName":"","Line":0},
"userlock_entry":{"UserNum":8,"UPID":7480,"Tenant":0,"DomainID":-4,"UserName":"richb",

 "TTYDevice":"/dev/pts/9","CurrLocks":0,"LockHWM":1142,"ProgName":"","Line":0},
"userlock_entry":{"UserNum":9,"UPID":7486,"Tenant":0,"DomainID":-4,"UserName":"richb",

 "TTYDevice":"/dev/pts/10","CurrLocks":1,"LockHWM":1126,"ProgName":"","Line":0},
"userlock_entry":{"UserNum":10,"UPID":7490,"Tenant":0,"DomainID":-4,"UserName":"richb",

 "TTYDevice":"/dev/pts/11","CurrLocks":0,"LockHWM":1116,"ProgName":"","Line":0}},
"locktable_detail":{
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":6581,"Partition#":0,"HashChain":398,

 "LockType":7492,"LockType":6,"LockType":0,"LockType":-4,"LockType":"S   L       ",

"LockType":"richb","LockType":"/dev/pts/12","UserNum":"BEGIN","Tenant":"FWD","DomainID":251702},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":6581,"Partition#":0,"HashChain":398,

 "LockType":7486,"LockType":9,"LockType":0,"LockType":-4,"LockType":"S   L       ",
```

```
"LockType":"richb","LockType":"/dev/pts/10","UserNum":"BEGIN","Tenant":"FWD","DomainID":251704},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":7539,"Partition#":0,"HashChain":119,

 "LockType":7492,"LockType":6,"LockType":0,"LockType":-4,"LockType":"S           ",

"LockType":"richb","LockType":"/dev/pts/12","UserNum":"BEGIN","Tenant":"FWD","DomainID":251702},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":6583,"Partition#":0,"HashChain":400,

 "LockType":7486,"LockType":9,"LockType":0,"LockType":-4,"LockType":"S   L       ",

"LockType":"richb","LockType":"/dev/pts/10","UserNum":"BEGIN","Tenant":"FWD","DomainID":251704},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":6579,"Partition#":0,"HashChain":396,

 "LockType":7492,"LockType":6,"LockType":0,"LockType":-4,"LockType":"S   L       ",

"LockType":"richb","LockType":"/dev/pts/12","UserNum":"BEGIN","Tenant":"FWD","DomainID":251702},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":7539,"Partition#":0,"HashChain":119,

 "LockType":7486,"LockType":9,"LockType":0,"LockType":-4,"LockType":"S           ",

"LockType":"richb","LockType":"/dev/pts/10","UserNum":"BEGIN","Tenant":"FWD","DomainID":251704},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":6579,"Partition#":0,"HashChain":396,

 "LockType":7486,"LockType":9,"LockType":0,"LockType":-4,"LockType":"S   L       ",

"LockType":"richb","LockType":"/dev/pts/10","UserNum":"BEGIN","Tenant":"FWD","DomainID":251704},
"locktable_detail":{"LockType":"REC ","Table
#":2,"Rowid":6583,"Partition#":0,"HashChain":400,
 "LockType":7492,"LockType":6,"LockType":0,"LockType":-4,"LockType":"S   L       ",

"LockType":"richb","LockType":"/dev/pts/12","UserNum":"BEGIN","Tenant":"FWD","DomainID":251702},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":7540,"Partition#":0,"HashChain":120,

 "LockType":7492,"LockType":6,"LockType":0,"LockType":-4,"LockType":"S   L       ",

"LockType":"richb","LockType":"/dev/pts/12","UserNum":"BEGIN","Tenant":"FWD","DomainID":251702},
"locktable_detail":{"LockType":"REC","Table#":2,"Rowid":7538,"Partition#":0,"HashChain":118,

 "LockType":7492,"LockType":6,"LockType":0,"LockType":-4,"LockType":"S   L       ",

"LockType":"richb","LockType":"/dev/pts/12","UserNum":"BEGIN","Tenant":"FWD","DomainID":251702}},
```


Syserror event files

When a syserror event is triggered, the output file is created in the event subdirectory is the same for all levels. The output is identical to the file created for the lock table summary file, and re-uses the lock table summary file name.

For details on the lock table summary file, see Lock table overflow files on page 409.

20

Dumping and Loading

Dumping and reloading data definitions and table contents is important for both application development and database maintenance. This chapter details the different ways you can perform a dump and load.

Note: For a complete description of dumping and loading SQL content, see the SQLDUMP and SQLLOAD descriptions in SQL Utilities on page 849

For details, see the following topics:

- Overview of dumping and loading
- Dumping ABL database definitions
- Dumping Change Data Capture policies
- Loading Change Data Capture policies
- Dumping database contents
- Loading database definitions
- Loading database contents
- Bulk loading
- Reconstructing bad load records
- Specialized dump and load techniques

Overview of dumping and loading

You might want to dump and load your database to:

- Create a new version of a database
- Use a constant table in multiple databases
- Economize disk space
- Convert a database to a new OpenEdge release or a different operating system platform
- Load updated data definitions to upgrade a database schema

When you dump a database, you must first dump the database or table definitions, and then dump the table contents. The definitions and contents must be in separate files and cannot be dumped in one step. You perform both procedures with the Data Administration tool if you are using a graphical interface, or the Data Dictionary if you are using a character interface.

You can also dump and load the table contents with PROUTIL. This option dumps and loads the data in binary format.

OpenEdge Data tools automatically attempt to disable triggers before performing the dump or load. If you do not have Can-Dump and Can-Load privileges, the Data tool asks if you want to dump or load the database without disabling the triggers. See Maintaining Security on page 169 for information about assigning privileges.

For information on manually dumping and loading database contents using ABL syntax, see the entries for `EXPORT` and `IMPORT` in *OpenEdge Development: ABL Reference*.

Dump and load limitations

There are three restrictions for using the Dump and Load option from the OpenEdge Data tools:

- When you reload data from one database to another, the reload must be to a target database that has the **same** fields, arranged in the **same** logical order as the source database. The schema for each database must be identical. Otherwise, you must write your own reload procedure using such ABL statements as `INPUT FROM`, `CREATE`, and `IMPORT`.
- If you define a database field with a data type of ROWID or RECID, then the ROWID values in that field are dumped and reloaded as `Unknown value (?)`. You must write your own dump and reload procedures to accommodate ROWID fields.
- If you do not have Can-Create and Can-Write privileges for a file, you can still dump the data and data definitions for that file. You can also load the data definitions for the file, but you cannot load the data for that file.

Note: OpenEdge RDBMS table data can also be dumped and loaded in formats other than those described in this chapter.

Dumping ABL database definitions

Use the Data Dictionary or Data Administration tool to dump ABL (Advanced Business Language) database definitions. There are two ways to dump the database definitions:

- Dump the entire database, including all its tables and fields

- Dump individual tables, sequences, or auto-connect records

Whenever you run the dump utility to dump table definitions, the Data Dictionary or Data Administration tool creates a data definitions (`.df`) file that contains definitions of tables, fields, indexes, sequences, and auto-connect records, and all their characteristics. However, depending on whether you choose to dump all tables or only selected tables when you dump the definitions, the Data Dictionary or Data Administration tool might or might not write all definitions to the `.df` file.

The table below shows the definitions that are dumped in each case.

Table 52: Definitions dumped to the definition file

Definitions	All tables	Selected tables
Tables, fields, and indexes	Yes	Yes
Sequence definitions	Yes	No
Auto-connect records	Yes	No
Collation	No	No
_User	No	No

If you dump individual tables, you must also dump the sequence definitions and auto-connect records separately. See Dumping sequence definitions on page 416 and Dumping auto-connect records on page 416 for instructions.

Dumping ABL database or table definitions

Use the following procedure to dump ABL database or table definitions:

1. Access the Data Administration tool. The Data Administration main window appears.
2. Make sure you are using the database (the source) from which you want to dump the table definitions.
3. Choose **Admin** > **Dump Data and Definitions** > **Data Definitions (.df file)**. The Data Administration tool lists all the tables defined for the database alphabetically.
4. Choose the table where you want to dump the definitions, or choose `ALL`.

   If you choose `ALL`, the `.df` file will also contain the sequences and auto-connect record definitions, but not the collation/conversion table. The Data Administration tool displays a default name for the file that you can dump data definitions into (hidden tables are not dumped). This default file is always the name of the table or database with a `.df` extension. The Data Administration tool truncates table names to eight characters. When you dump only one table, the table dump name becomes the default for its corresponding contents dump file. For example, if you specify `customer.d` as the filename, when you dump the file contents, the default filename is `customer.d`. If you specify to dump all the tables, the default name is *db-name*`.d`.

5. Accept this default or enter a different name and choose `OK`. The Data Administration tool displays each object name as it writes its definition to the data definitions file.

   You see each table name on the screen as the Data Administration tool writes its definition to the `.df` file. After all of the table definitions are dumped, the Data Administration tool displays a status message and prompts you to continue.

6. Choose `OK` to return to the Data Administration main window.

Definition file format

When you dump definitions, the Data Dictionary or the Data Administration tool creates a data definitions file. The Data tool then uses the data definitions file to load table definitions into the database. The following figure shows a sample data definitions file for a single table.

Figure 52: Sample data definitions file

```
ADD TABLE "Customer"
  AREA "Info Area"
  DESCRIPTION "Customer information"
  DUMP-NAME "customer"
  TABLE-TRIGGER "CREATE" NO-OVERRIDE PROCEDURE "sports/crcust.p" CRC "?"
  TABLE-TRIGGER "DELETE" NO-OVERRIDE PROCEDURE "sports/delcust.p" CRC "?"
  TABLE-TRIGGER "WRITE" NO-OVERRIDE PROCEDURE "sports/wrcust.p" CRC "?"

ADD FIELD "Cust-Num" OF "Customer" AS integer
  FORMAT ">>>>9"
  INITIAL "0"
  POSITION 2
  MAX-WIDTH 4
  VALEXP "cust-num > 0"
  VALMSG "Customer number must be greater than zero"
  ORDER 10

ADD FIELD "Name" OF "Customer" AS character
  FORMAT "x(20)"
  INITIAL ""
  POSITION 3
  MAX-WIDTH 40
  ORDER 30

ADD FIELD "Address" OF "Customer" AS character
  FORMAT "x(20)"
  INITIAL ""
  POSITION 4
  MAX-WIDTH 40
  ORDER 40
 .
 .
 .

ADD INDEX "Cust-Num" ON "Customer"
  AREA "Info Area"
  UNIQUE
  PRIMARY
  INDEX-FIELD "Cust-Num" ASCENDING

ADD INDEX "Comments" ON "Customer"
  AREA "Info Area"
  WORD
  INDEX-FIELD "Comments" ASCENDING
 .
 .
 .

.
PSC
cpstream=ISO8859-1
.
0000003848
```

If you modify a data definitions file, tables, fields, and indexes can be intermixed within one `CREATE DATABASE` section. However, be sure to adhere to the following:

- Define a file before any fields contained in that file
- Define a field before any index using that field
- Specify information using the correct syntax
- All definitions must belong with the most recent `ADD DATABASE` or `CREATE DATABASE` entry in the `.df` file

The trailer information contains the values used by the database definitions. If you edit the definitions file or create one manually, be sure to specify these values correctly. The table below explains these values.

Table 53: Data definitions file trailer values

Field	Description
`codepage=` *codepage*	The code page used by the database. For more information, see *OpenEdge Development: Internationalizing Applications*.
`encpolicy=yes`	Indicates that there are encryption policy settings in the data definition file.
`bufpool=yes`	Indicates that there are Alternate Buffer Pool settings in the data definition file.
Character count	The number of characters contained in the contents file, up to and including the period that indicates the end of the contents. This number is always a 10-digit number with leading zeros, if necessary.

Multi-tenant keywords

If you are dumping a multi-tenant database, multi-tenant tokens are present if the field is true. Multi-tenant tokens are:

- `MULTITENANT=yes`

  `MULTITENANT=yes` identifies whether the table or sequence is multi-tenant. This option is allowed with the `ADD` and `MODIFY` operation keywords for a table, and with the `ADD` operation keyword for a sequence.

  Note:

  Loading a `.df` file with this token gives an error if loaded in a database that is not multi-tenant enabled.

- `NO-DEFAULT-AREA` `NO-DEFAULT-AREA` indicates that the "keep default area" of the user interface (Data Admin) is false. `NO-DEFAULT-AREA` is specified on a multi-tenant enabled table k and is only allowed with the `ADD` operation keyword.

Creating an incremental ABL data definitions file

An incremental data definitions file is a data definitions file that contains the schema differences between two databases. When you load the incremental data definitions file into a target database, the target database will have the same schema as the source database. For example, you might use this to upgrade a production database to the same schema as a development database.

Note: If the data type of the extent of a field has changed, the Data tool writes a DROP statement to the .df file to delete the existing field, and then an ADD statement to re-create the field with its new definition. When the field is deleted from the database, the contents of that field are also deleted. To save the contents of the field, save the data before running this procedure, or rename the old field, create a new field separately, and write an ABL procedure that transfers the data from the old field to the new field. Then use the incremental dump procedure to update the rest of the schema, including deleting the obsolete old field.

To create an incremental data definitions file:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Select the source database as the current working database. The source database is the database that has the schema that you want the definitions file to have.
3. Choose **Admin** > **Dump Data and Definitions** > **Create Incremental .df File**. The Data tool lists all the connected databases.
4. If you have more than two databases connected, choose the target database. The target database is the database to update. The Data tool prompts you for the file where you want to write the differences.
5. Specify the file where you want to write the differences. The default filename is delta.df. The Data tool displays the file, field, sequence, and index names as it compares the databases.
6. After comparing the database, the Data Administration tool or the Data Dictionary returns you to the main window.

Note: If you use this option to create a .df file in conjunction with r-code files to update schema changes, you must load the .df file and recompile before you can run the new r-code. You must recompile because the Data tool reorders the indexes during the dump and load procedure.

Dumping sequence definitions

You can dump sequence definitions and values using either Data tool. The tools store the sequence definitions in the full database schema definition file that you name, and the sequence values in a separate _seqvals.d file (or the file you name). You can dump definitions and values separately.

To dump sequence definitions:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure that the database containing the sequences you want to dump or load is the current working database.
3. Choose **Admin** > **Dump Data and Definitions** > **Sequence Definitions**. The Data tool prompts you for the file where you want to write the sequence definitions. The default filename is _seqdefs.df.
4. Specify the filename or use the default value. After dumping the sequence definitions, the Data tool displays a status message and prompts you to continue.
5. Choose OK to return to the Data Administration or Data Dictionary main window.

Dumping auto-connect records

Dumping auto-connect records is similar to dumping table and sequence definitions.

To dump auto-connect records:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Choose **Admin** > **Dump Data and Definitions** > **Auto-connect Records Only**. The data tool prompts you for the output file where you want to write the auto-connect records. The default filename is `_auto.df`.
3. Specify a new filename or accept the default. After dumping the auto-connect records, the Data tool displays a status message and prompts you to continue.
4. Choose `OK` to return to the Data Administration or Data Dictionary main window.

Dumping Change Data Capture policies

You can dump Change Data Capture (CDC) policies from your database into a `.cd` file using the Data Administration tool.

To dump CDC policies using the Data Administration tool:

1. In the Data Administration window, click **Admin** > **Dump Data and Definitions** > **Change Data Capture Policies (.cd) file**.

   The **Select Change Data Capture Policies for Dump** window appears with a list of current, pending, and previous CDC policies available for the connected database. The CDC policies are alphabetically sorted by their names.

   Note: The **Change Data Capture Policies (.cd file)** option is disabled for databases that are not enabled for CDC.

2. From the list, select the CDC policies that you want to dump.

   When selecting the CDC policies, you can:

   - Select an individual CDC policy.
   - Select the **All Current** check box to select all current CDC policies.
   - Select the **All Pending** check box to select all pending CDC policies.

   Note: When dumping multiple policies, you must specify only one policy per source table as the Data Administration tool does not allow selecting more than one policy per source table.

   Note: When dumping a previous policy, note that it is stored as a pending policy in the `.cd` file.

3. Click **OK**.

   The **Dump Change Data Capture Policies** window appears.

4. In the **Output File** field, specify a name for the `.cd` file or use the default name *db-name*`.cd`.
5. To change the default directory for the output file, click **Files** and then browse to the destination directory where you want to store the file.
6. In the **Code Page** field, specify the code page format in which you want the CDC policies to be dumped or use the default format `ISO8859-1`.
7. Click **OK**.

A warning message appears if the code page differs from the database code page value. Click **OK** to continue. The Data Administration tool displays each item as it dumps the CDC policies, and a status message appears.

8. Click **OK** to complete the procedure.

Loading Change Data Capture policies

To load CDC policies using the Data Administration tool:

1. In the Data Administration window, click **Admin** > **Load Data and Definitions** > **Change Data Capture Policies (.cd file)**.

   The **Load Change Data Capture Policies** window appears.

2. In the **Input File** field, specify the file name or click **Files** to browse and select the `.cd` file that you want to load.
3. In the **Acceptable Error Percentage** field, type the error rate that is acceptable to the Data Administration tool to load a `.cd` file into a database.

   For example, if you specify an error rate of 10 percent, the Data Administration tool must successfully load 90 records from every set of 100 records. If the error rate exceeds, it terminates the loading process and places the bad records in the error `.e` file.

4. Click **OK**.

   The Data Administration tool displays each item as it loads the CDC policies, and a status message appears.

   Note: If the `.cd` file does not specify a code page, the Data Administration tool prompts you to specify the code page format.

5. Click **OK** to complete the procedure.

Dumping database contents

Once you dump the database's definitions, you dump the contents of each database object. When you dump database contents, you can dump the following:

- Table contents
- Sequence values
- User table contents
- SQL View File contents

Note: For a complete description of dumping and loading SQL content, see SQL Utilities on page 849

The OpenEdge RDBMS provides two methods of dumping database contents: you can use the PROUTIL utility to dump the data in binary format, or you can use the Data Administration tool user interface to dump the data in text format. Dumping the data in binary format with PROUTIL is generally faster.

Dumping table contents with PROUTIL

PROUTIL DUMP allows you to perform a binary dump of data from an online or offline database. Online binary dumps can be multi-threaded; however, multi-threading requires an Enterprise database license. The records will be dumped in order according to an index. If you do not specify an index, the table's primary index is used. You must specify the index by its number. You can use the PROUTIL IDXANALYS utility to help determine the index number. For the complete syntax of the PROUTIL IDXANALYS utility, see PROUTIL IDXANALYS qualifier on page 712.

Use the following syntax to perform an offline or single-threaded online binary dump:

Syntax

```
proutil db-name -C dump [owner-name.] table-name directory [-index num]
```

In the syntax, *db-name* specifies the database from which you want to dump; *owner-name* specifies the owner of the table containing the data you want to dump; *table-name* specifies the name of the table containing the data you want to dump; *directory* specifies the name of the target directory where the data will be dumped; and -index *num* specifies an index to use to dump the table's contents.

Expand the PROUTIL DUMP syntax for a threaded online dump with the following options:

```
{ [-thread n] [-threadnum nthreads] [-dumplist dumpfile] }
```

Where -thread 0 indicates a single-threaded dump, and -thread 1 indicates a multi-threaded dump; -threadnum*nthreads* specifies the maximum number of threads to create; and -dumplist *dumpfile* creates a list of the dump files generated. For complete description of the PROUTIL DUMP syntax, see PROUTIL DUMP qualifier on page 689.

The number of threads created during a threaded dump is capped at the value specified by -threadnum. If you do not specify a value, the default maximum is the number of system CPUs. The actual number of threads created may be less than the maximum. PROUTIL DUMP determines the optimal number of threads to create based on the complexity of the index specified.

Results of a binary dump with PROUTIL

PROUTIL DUMP writes data from a table to a dump file or files. The name of the resulting dump files depends on the owner of the table. When tables owned by PUB are dumped to a single file, the filename is the table name with .bd appended. For example, *tablename*.bd.

However, when tables owned by anyone other than PUB are dumped to a file, the resulting filename contains the owner name and table name. For example, *ownername_tablename*.bd.

On systems that have a 2GB file size limitation, a single-threaded PROUTIL DUMP creates multiple files when you dump a table larger than 2GB. For example, when you dump data from a table with the name customer that is 6.4GB, PROUTIL DUMP creates four binary dump files: customer.bd,customer.bd2, and customer.bd3, each of which is approximately 2GB, and customer.bd4, which is approximately 0.4GB. The PROUTIL DUMP procedure adds header blocks to the binary dump files. As a result, the total size of the binary dump files is slightly larger than the table itself.

On systems without file size limitation, a single-threaded PROUTIL DUMP creates only one binary dump file regardless of the size of the table.

Multi-threaded PROUTIL DUMP creates a file for each thread created. The first thread creates the file, *tablename*.bd; the second thread creates the file *tablename*.bd2; each additional thread creates a file with an incremental number, *tablename*.bd*n*. Use the -dumpfile option to generate a list of the files created by the threaded dump. The file specified by the -dumpfile option contains a list of fully qualified file names, and can be used as input to PROUTIL LOAD. If the file specified by the -dumpfile option exists, PROUTIL DUMP will overwrite the existing file.

Format of a binary dump file

Each binary dump file contains a header and a description for each record in the table. The dump file appears in the following format:

Header	Record length	Table number	Binary record	Record CRC

The file header contains information that appears in the following order:

1. Version number
2. Date and time the file was created
3. Name of the table being dumped
4. Number of the table being dumped
5. CRC of the table being dumped
6. Number of fields in the table
7. Name of the database where the table resides
8. Section number
9. Number of the first record
10. Table owner

Section numbers that appear in the file header correspond to the multiple binary dump files created for a single table by a multi-threaded dump or by a single-threaded dump of a table that exceeds the 2GB file size limitation. For example, Section 1 corresponds to the binary dump file named customer.bd, Section 2 corresponds to customer.bd2, Section 3 corresponds to customer.bd3, and Section 4 corresponds to the customer.bd4 binary dump file.

Note: To support the dump and load of binary large objects (BLOBS) and character large objects (CLOBS), PROUTIL DUMP adds more items to the header of the binary dump file.

Dumping selective contents with PROUTIL

A selective binary dump allows you to specify that only records matching your criteria are dumped.

Use the following syntax to perform a selective binary dump within a bracketed range:

```
proutil db-name -C dumpspecified [owner-name.] table-name.field-name
   operator1 low-value AND operator2 high-value directory
        [-preferidx index-name]
```

The table below describes selective dump syntax.

Table 54: DUMPSPECIFIED syntax

Field	Description
db-name	Specifies the database from which you want to dump.
owner-name	Specifies the owner of the table containing the data you want to dump.
table-name	Specifies the name of the table containing the data you want to dump.
field-name	Specifies the index you are using to select the records to dump.
operator1 and *operator2*	Specifies the comparison to make when selecting records. Operator can be one of five values: • **EQ** —Equal to • **GT** — Greater than • **GE** — Greater than or equal to • **LT** — Less than • **LE** — Less than or equal to **Note:** The first value and *operator*, combined with the second value and *operator*, define a subset. You can not use AND to dump two sets of non-contiguous results.
low-value and *high-value*	Specifies the range of values against which the field contents will be compared.
AND	Specifies a bracketed range for the binary dump.
directory	Specifies the name of the target directory where the data will be written.
-preferidx *index-name*	By default, DUMPSPECIFIED uses the primary index to sort rows; specifying an index with -preferidx orders the rows by *index-name*.

Note: For databases enabled for table partitioning or multi-tenancy, you can further refine your dump by partition, tenant, or group. For details on the complete syntax, see PROUTIL DUMPSPECIFIED qualifier on page 692.

Examples

The following syntax dumps all order date values greater than 2/3/02 from the Sports2000 database:

```
proutil sports2000 -C dumpspecified order.order_date GT '02-03-2002'
```

The following syntax dumps all item prices less than $25.90 from the Sports2000 database:

```
proutil sports2000 -C dumpspecified item.price LT 25.9
```

The following syntax dumps all order ship dates of 12/24/01 from the Sports2000 database into the `/dev/results` directory:

```
proutil sports2000 -C dumpspecified order.ship_date EQ '12-24-2001' /dev/results
```

The following syntax dumps all back ordered items from the Sports2000 database into the `/inventory/warehouse1` directory:

```
proutil sports2000 -C dumpspecified order_line.backorder EQ yes /inventory/warehouse1
```

Dumping table contents with a Data tool

When you dump table data using the Data Administration or Data Dictionary tools, the Data tool creates a text contents file that contains data from each table you specify. This file has a `.d` extension. The Data tool uses the contents file to load data into the database. The Data tool creates a separate contents file for every table you dump.

To dump table contents:

1. Access the appropriate Data tool (Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure you are using the database (the source) where you want to dump the table data.
3. Choose **Admin** > **Dump Data and Definitions** > **Table Contents (.d files)**. The Data tool lists all the tables defined for the database alphabetically.
4. Mark the table contents you want to copy. You can mark tables with asterisks (*), then use the **Select Some** and **Deselect Some** buttons to select or deselect groups of tables. If you are using a character interface, use **F5** and **F6**.

   When you dump a single table, the Data tool displays a default name for the file that you can dump the table contents into (hidden tables are not dumped). This default file is always the dump name of the table definition file, with a `.d` extension. If you want to specify a file other than the default file, type the name in the **Output file:** field.

   When you dump table contents for more than one table, the Data tool prompts you for a directory name in the **Output Directory** field. If you do not specify a directory, the Data tool dumps the table contents files into the current directory. The Data tool names each contents file with its corresponding table name.

   Accept the default or enter a different name.
5. If your database contains LOBs you want dumped, check the **Include LOB:** check box.
6. If you checked the **Include LOB:** check box, specify the location of the LOBs in the **LOB Directory:** field.
7. If you want to use character mapping, enter the character mapping, then choose **OK**. See *OpenEdge Development: Internationalizing Applications* for information about character mapping, PROTERMCAP, and national language support.

The Data tool displays each table name as it writes the contents to the table contents file. After dumping the contents, the tool displays a status message and prompts you to continue.

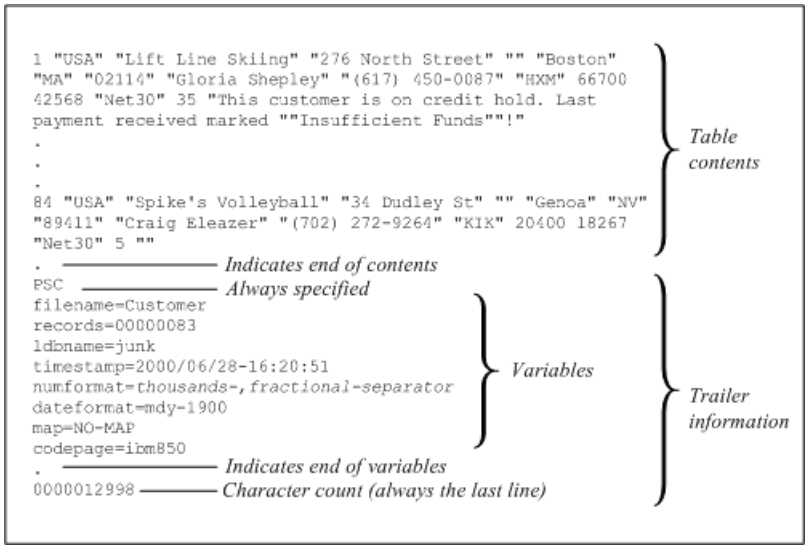
8. Choose OK to return to the Data Administration or Data Dictionary main window.

Note: You must dump the sequence values separately. See Dumping sequence values with a Data tool on page 424 for more information.

Contents file format

The following figure shows a sample contents file.

Figure 53: Sample contents file



The trailer information contains information about the source database and certain startup parameters specified for the session in which the contents file was created. Certain variables are included for informational purposes only; other variables are used to load the data correctly. If you edit the contents (.d) file or create one manually, be sure to specify these variables correctly.

The table below explains these variables. If the database uses the default values for these variables, these variables do not appear in the contents file.

Table 55: Contents file trailer variables

Field	Description
filename= *table-name*	Table name.
records= *num-records*[9]	Number of records contained in the file.
ldbname= *logical-db-name*	Logical database name of the source database.
time stamp= *time stamp*	Time stamp of the database.

[9] Information used when loading the file. If the value is specified in the trailer and is specified incorrectly, the data will not load.

Field	Description
num format=[9] *thousands-separator, fractional-separator*	The numeric value of the character that represents the thousands-separator and fractional-separator identified in the code page. When you attempt to load the table contents file, the session and file must use the same numeric format. If not, OpenEdge reports a run-time error.
dateformat= *date-format*[9]	The date format used by the database contents.
map={MAP *protermcap-entry* | NO-MAP}[9]	The character mapping to support the MAP option on the INPUT FROM and OUTPUT TO statements. You can specify a mapping between a standard set of ASCII (7-bit) characters and extended (8-bit) characters in the PROTERMCAP file. TROTERMCAP entries are used to build a translation table for the stream. MAP specifies an entry that has been defined in the PROTERMCAP file. If NO-MAP is specified, character translation is bypassed.
codepage= *codepage*	The code page used by the database. For more information, see *OpenEdge Development: Internationalizing Applications*.
Character count	The number of characters contained in the contents file, up to and including the period (.) that indicates the end of the contents; this number is always a ten-digit number with leading zeros, if necessary.

Dumping sequence values with a Data tool

The procedure for dumping sequence values is similar to dumping table contents.

To dump sequence values:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure that the database containing the sequences you want to dump or load is the current working database.
3. Choose **Admin** > **Dump Data and Definitions** > **Sequence Values**. The Data tool prompts you for the file you want to write the sequence values to. The default filename is _seqvals.d.
4. Specify the filename or use the default value. After dumping the sequence values, the Data tool displays a status message and prompts you to continue.
5. Choose OK to return to the Data Administration or Data Dictionary main window.

Dumping user table contents with a Data tool

The procedure for dumping user table contents is similar to the procedures for dumping table contents and sequence values.

To dump user table contents:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure you are using the database from which you want to dump the table data.
3. Choose **Admin** > **Dump Data and Definitions** > **User Table Contents**. The Data tool prompts you for the file to write the user file contents to. The default filename is `_user.d`.
4. Specify the filename or accept the default. After dumping the user file contents to the specified file, the Data tool displays a status message and prompts you to continue.
5. Choose `OK` to return to the Data Administration or Data Dictionary main window.

Dumping an SQL view file's contents with a Data tool

The procedure for dumping the contents of an SQL file is similar to the procedures for dumping table contents and sequence values.

To dump the contents of a SQL view:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure you are using the database (the source) where you want to dump the table data.
3. Choose **Admin** > **Dump Data and Definitions** > **Views**. The Data tool prompts you for the file, then for the file that you want to write the user file contents to. The default filename is `_view.d`.
4. Specify the filename or accept the default. After dumping the view file contents to the specified file, the Data tool displays a status message and prompts you to continue.
5. Choose `OK` to return to the Data Administration or Data Dictionary main window.

Note: For a complete description of dumping and loading SQL content, see SQL Utilities on page 849.

Loading database definitions

You can load either all the data definitions for a table or only those definitions that have changed.

The Data tools use a data definitions file that you specify to load table definitions into the database. A data definitions file contains definitions for tables, fields, and indexes.

To load table definitions into a database:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure you are using the database (the target) where you want to load the table definitions.
3. Choose **Admin** > **Load Data and Definitions** > **Data Definitions (.df files)**. The Data tool prompts you for the name of the file that contains the data definitions you want to load into the current database. The default filename is the logical name of the current working database, with a `.df` extension.
4. Specify a file or accept the default.
5. Specify whether you want to stop the load as soon as the Data tool encounters a bad definition statement. The Data tool displays each item as it loads the definitions for that object. The Data tool displays a status message and prompts you to continue. If you choose not to stop on errors, the load will continue with the next entry.

Note: Whatever you choose, if the Data tool encounters any error, it backs out any loaded definitions.

6. Choose OK to return to the Data Administration or Data Dictionary main window.

The database now contains the table, field, index, or sequence, but none of the data.

Loading updated ABL data definitions

You can update an existing database schema to include schema changes made in a new database. Note that this procedure can also be used to merge two databases.

To load updated ABL data definitions:

1. Make a copy of the database you want to update and save the original. The database should not be empty.
2. Connect to the database that includes the new, modified data definitions.
3. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
4. Choose **Database** > **Connect Database**. The **Database Connect** dialog box appears.
5. Enter the name of the database you want to connect to and choose OK. The Data tool connects the database and returns you to the Data tool's main window.
6. Choose **Admin** > **Dump Data and Definitions** > **Create Incremental .df File**. The **Create Incremental .df File** dialog box appears.

   The **Create Incremental .df File** option compares the data definitions in the nonempty copy to the current database schema and creates a new data definitions file. The new `.df` file contains a record for each difference between the two schemas. The differences include any added, renamed, changed, or deleted table, field, or index.

   If a table, field, or index exists in the old database but not in the new schema, the Data tool asks if you renamed the object. If you answer no, a record appears in the new `.df` file marking the object as deleted.

   If the new schema includes a new, unique, active index, the Data tool prompts you to deactivate it. If you do not deactivate the index and there are duplicate keys in the old database, the system aborts your attempt to load new definitions into the old database. If you deactivate the index, the load procedure defines the new index but does not create the index file. You must complete Step 8 to build and activate the index after loading the new data definitions.

7. Enter the database name or accept the default databases, then choose OK.
8. Connect to the copy of the old database.
9. Load the updated data definitions by choosing **Admin** > **Load Data and Definitions** > **Data Definitions (.df files)**.
10. If you deactivated any indexes, re-create data in the indexed fields as required to avoid duplicate keys, then reactivate the indexes with PROUTIL IDXBUILD.
11. The Data tool updates the old database schema to match the modified schema. Compile and test all your procedures against the updated database.

Loading database contents

These sections describe how to load the following types of contents:

- Table contents

- User table contents
- SQL file view contents
- Sequence values

Note: For a complete description of dumping and loading SQL contents, see SQL Utilities on page 849.

The OpenEdge RDBMS provides three methods of loading table contents. You load data dumped in binary format with the PROUTIL LOAD command. Data in text format is loaded with a Data tool's user interface or the PROUTIL BULKLOAD command. You can perform a binary load only on database contents that were created with a binary dump.

Loading table contents in binary format with PROUTIL

Use the following syntax to perform a binary load:

```
proutil db-name -C load filename [ -dumplist dumpfile]
```

In the syntax, *db-name* specifies the database where you want to load the data. To load one file, specify *filename*; to load multiple binary dump files, specify -dumplist *dumpfile* where *dumpfile* contains a list of binary dump files. You can chose to build indexes on your data as it loads with the `build indexes` parameter. For complete syntax details, see PROUTIL LOAD qualifier on page 739.

When specifying multiple files to load, you can use a dump file created with a multi-threaded binary dump, or create your own. A dump file contains a list of fully qualified file names of binary dump files. This example shows the contents of the file `order.dmp_lst`:

```
/usr1/docsample/101A/bindump/order.bd
/usr1/docsample/101A/bindump/order.bd2
/usr1/docsample/101A/bindump/order.bd3
/usr1/docsample/101A/bindump/order.bd4
/usr1/docsample/101A/bindump/order.bd5
/usr1/docsample/101A/bindump/order.bd6
```

To load all the files listed in `order.dmp_lst` into a database named `newdb`, issue the following command:

```
proutil newdb -C load filename -dumplist order.dmp_lst
```

Using the `-dumplist` parameter is the equivalent of issuing 6 individual load commands. For example:

```
proutil newdb -C load /usr1/docsample/101A/bindump/order.bd
proutil newdb -C load /usr1/docsample/101A/bindump/order.bd2
proutil newdb -C load /usr1/docsample/101A/bindump/order.bd3
proutil newdb -C load /usr1/docsample/101A/bindump/order.bd4
proutil newdb -C load /usr1/docsample/101A/bindump/order.bd5
proutil newdb -C load /usr1/docsample/101A/bindump/order.bd6
```

When the load procedure finishes, it reports the number of records that were loaded.

Recovering from an aborted binary load

The procedure to recover from an aborted binary load depends on how much the binary load completed before the abort. If the binary load aborted during the sort phase (after all the rows loaded), then recovery entails running the index rebuild utility on the affected table. If the binary load aborted during the load of the rows into the database, there is a separate procedure for recovery.

To recover from a binary load that aborted during the loading of rows:

1. Truncate the storage area.
2. Restart the load operation.

You can determine which phase the binary load completed by examining the database log (`.lg`) file. In this file, you will see messages identifying the major phases of the load operation, such as:

- Binary load session started
- Record load phase started
- Record load phase completed
- Starting index build phase
- Index build phase completed
- Binary load session ended

Loading table contents with a Data tool

ABL Data tools use a contents file that you specify to load table data into the database. A contents file contains all the data for a table.

Note: Table definitions must be in the database before you can load table contents.

To load table contents into a database:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure that the working database is the target database where you want to load the table contents.
3. Choose **Admin** > **Load Data and Definitions** > **Table Contents (.d files)**. The Data tool alphabetically lists all the tables defined for the database.
4. Mark the tables whose contents you want to copy. Use the **Select Some** and **Deselect Some** buttons to select or deselect groups of tables. For a character interface, use **F5** and **F6** to select table names.

   The Data tool prompts you for the name of the contents file (or the directory that contains the contents files) you want to load into the current database.

5. Specify the directory name, filename, or use the default value.

   When you load the contents of an entire database, enter a directory where you want to load the contents files. If you do not specify a directory, the Data tool loads the files from the current directory. The Data tool loads each table from the corresponding *table-dumpname*`.d` filename. If you choose a single table to load, the Data tool displays a default name for the file where you can load the table contents. This default file is always the name of the table definition file, with a `.df` extension.

6. Specify if you are loading LOBs by checking **Include LOB**.

7. If you are including LOBs, specify the directory in the **Lob Directory** field. If you don't specify a directory, the current directory is assumed.
8. Specify an acceptable error rate.

   As the Data tool loads records from any files you designate, it might encounter data that cannot be loaded. For example, a record might have data in one of its fields that is too long for that field's defined format. The Data tool does not load the record. If you specify an error rate of 10 percent, the Data tool must successfully load 90 records from every set of 100 records loaded. If you use the graphical interface to load table contents, you can choose to output the errors to a file or output the errors to the screen. After loading the table contents, the Data tool prompts you to continue.
9. Check **Output Errors to Screen**, if you want errors reported to the screen as they occur.
10. Choose `OK` to return to the Data Administration or Data Dictionary main window.

Note: You must load the sequence values separately. See Loading sequence values using a Data tool on page 430 for information about loading sequence values.

Loading user table contents with a Data tool

The process for loading user table contents is similar to that for loading table contents.

To load the user table of contents:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface, or the Data Dictionary if you are using a character interface).
2. Choose **Admin** > **Load Data and Definitions** > **User Table Contents**. The Data tool prompts you for the file from where you want to read the user file contents. The default filename is `_user.d`.
3. Specify the filename or accept the default. After loading the user file contents to the specified file, the Data tool displays a status message and prompts you to continue.
4. Choose `OK` to return to the Data Administration or Data Dictionary main window.

Loading an SQL view file contents

The steps for loading the contents of an SQL view file are similar to those for loading table and user table contents.

To load an SQL view file's contents:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Choose **Admin** > **Load Data and Definitions** > **SQL Views**. The Data tool prompts you for an input file from which to load the SQL views. The default filename is `_view.d`.
3. Specify the filename or accept the default. After loading the view file contents from the specified file, the Data tool displays a status message and prompts you to continue.
4. Choose `OK` to return to the Data Administration or Data Dictionary main window.

Note: For a complete description of dumping and loading SQL content, see SQL Utilities on page 849.

Loading sequence values using a Data tool

The Data Administration tool and the Data Dictionary use a contents file that you specify to load sequence values into the database. Usually, this file is called `_seqvals.d`.

To load sequence values:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Make sure that the working database is the target database where you want to load the table contents.
3. Choose **Admin** > **Load Data and Definitions** > **Sequence Current Values**. The Data tool prompts you for the filename where you want to write the sequence values. The default filename is `_seqvals.d`.
4. Specify the filename or use the default value. After the sequence values are loaded, the Data tool displays a status message and prompts you to continue.
5. Choose `OK` to return to the Data Administration or Data Dictionary main window.

Bulk loading

The Bulk Loader utility loads text data at a higher speed than the Load utility provided with the Data Dictionary or Data Administration tool.

Note: The bulk loader works only with OpenEdge databases. Some non-OpenEdge databases offer a similar bulk loading tool. If you are using a non-OpenEdge database and a bulk loading tool is not available, you must use the standard load utility in the Data Administration tool or Data Dictionary.

Creating a Bulk Loader description file

You must create a bulk loader description file before you can run the Bulk Loader utility. You use the Data Administration tool or the Data Dictionary to create the description file. The default filename is *`table-dumpname`*`.fd`. If you choose more than one table, the Data tool adds the extension `.fd` to the name of the database. It then places all of the bulk loader information in that one file.

Note: For information on how to dump table data into contents files, see Dumping user table contents with a Data tool on page 424.

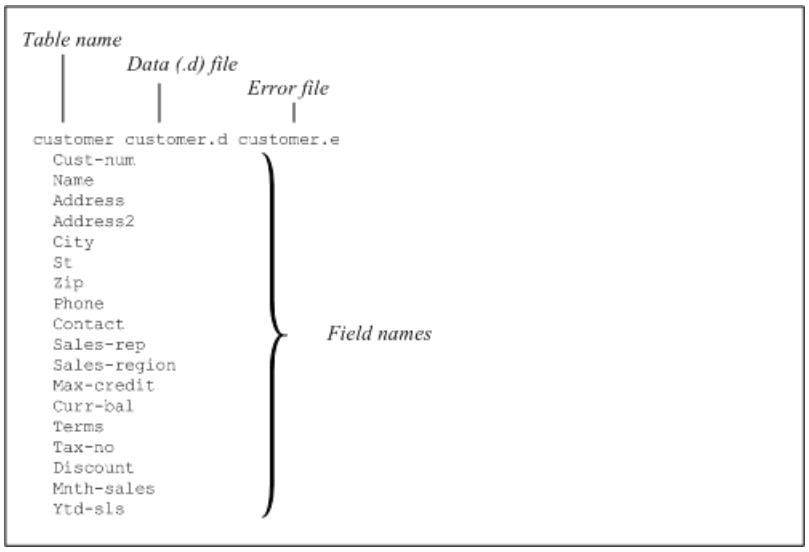
To create a bulk loader description file:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Choose **Admin** > **Create Bulk Loader Description File**. The Data tool alphabetically lists all the tables defined for the database.
3. Select the tables for which you want to create bulk loader description files. Use the **Select Some** and **Deselect Some** buttons to select or deselect groups of tables. For a character interface, use **F5** and **F6**. The Data tool prompts you for the bulk load filename. The default filename is *`table-dumpname`*`.fd`.
4. Specify the filename or accept the default.
5. If you are loading LOBs, check **Include LOB:** and specify their location in the **Lob Directory:** field.
6. Specify a code page or accept the default.

7. Click OK to create the bulk load description file. After creating the file, the Data tool displays a status message and prompts you to continue.
8. Choose OK to return to the Data Administration or Data Dictionary main window.

The following figure shows a description file created with the Data Administration tool.

Figure 54: Dictionary-created description file example



In this example, customer is the name of the database table, `customer.d` is the name of the data file you want to load, and `customer.e` is the name of an error file that the Bulk Loader creates if errors occur during the bulk load. The field names in the customer table are Cust-num, Name, Address, etc.

Modifying a Bulk Loader description file

You can use a text editor to change a Bulk Loader description (`.fd`) file. The table below lists the tasks and actions.

Table 56: Modifying the description file

Task	Action
Include comments in the `.fd` file.	Place a pound sign (#) at the beginning of each line.
Include the description of multiple tables in one description file.	Separate tables with a period on a line by itself.
Skip a field.	Replace the field name with a caret (^).

By default, the Bulk Loader adds `.d` and `.e` extensions to the table name to produce the two file names it requires to operate. It assumes that both of the files (the data file and the error file) begin with the table name. If this assumption is false, you must specify the different filenames.

For example, if you dump the customer table into `cust.d` instead of `customer.d`, you must specify the name `cust.d` in the bulk loader description file. Otherwise, the Bulk Loader searches the current directory for `customer.d`. The following figure shows a modified `.fd` file.

```
#This is an example
customer   cust.d cust.e
  Cust-num
  Name
  ^
  ^
   City
  St
  ^
  Phone
.
item
  Item-num
  Idesc
.
order
  Order-num
  ^
  Name
```

Loading table contents with the Bulk Loader qualifier

PROUTIL with the BULKLOAD qualifier allows you to load data tables into a database. Using contents files and bulk loader description (`.fd`) files, the Bulk Loader loads large amounts of data faster than the regular Data Administration or Data Dictionary Load utility.

The Bulk Loader can also load table data in formats other than contents files. For information about other file formats can load, see the `IMPORT` statement in *OpenEdge Development: ABL Reference*.

The Bulk Loader bypasses any `CREATE TRIGGER` statements and deactivates all indexes it encounters.

To load table contents with the Bulk Loader utility:

1. Create a Bulk Loader description file using the Data Administration tool, Data Dictionary, or a text editor. If you use the Data Dictionary or Data Administration tool, it automatically writes the description file. If you use a text editor, you must create the file.

   Using a text editor to modify a description file that the Bulk Loader has created allows you to customize the file to suit your needs. For more information see Creating a Bulk Loader description file on page 430.

2. Verify that the table definitions are in the database.
3. Run the Bulk Loader utility.

   The bulkloader syntax is:

   ```
   proutil db-name [ -yy n] -C bulkload fd-file [ -B n]
   ```

   Where *db-name* specifies the database you are using; `-yy`*n* indicates the start of a 100-year period in which any two-digit `DATE` value is defined; *fd-file* identifies the bulk loader description file; and -B *n* indicates the Blocks in Database Buffers startup parameter; *n* specifies the number of blocks.

   See PROUTIL BULKLOAD qualifier on page 647 for complete syntax details and more information about the PROUTIL BULKLOAD.

Note: On exceptionally small systems, you might encounter memory problems when using the Bulk Loader utility. To work around these problems, split the description file into several smaller files.

The Bulk Loader utility checks the description file to determine which data file contains the customer table's dumped records. In this example, it is `customer.d`.

The order of the fields in the description file must match the order of the fields in the data (`.d`) file. If they do not match, the Bulk Loader attempts to load data into the wrong fields.

The Bulk Loader utility automatically deactivates a data table's indexes, so you must run the Index Rebuild utility after the Bulk Loader loads the data files. If the Bulk Loader encounters duplicate key values during a load, it continues to load the data file's records. You must manually correct the data before you run the Index Rebuild utility.

Reconstructing bad load records

If the Database Administrator tool or the Data Dictionary encounters errors while loading a data file, it creates an error file. You can use the error file and the original data file to build a new data file for all the bad records.

To build a new data file for bad records:

1. Access the appropriate Data tool (the Data Administration tool if you are using a graphical interface or the Data Dictionary if you are using a character interface).
2. Choose **Admin** > **Load Data and Definitions** > **Reconstruct Bad Load Records**. The **Reconstruct Bad Load Records** dialog box appears.
3. Specify the original data file, the error file, and the new output file, then choose **OK**. The default filename for the new data file is `error.d`. After writing the new output file, the Data tool displays a status message and prompts you to continue.
4. Choose `OK` to return to the Data Administration or Data Dictionary main window.
5. Use a text editor to edit the new output file and fix the bad records after the Data tool builds the file. Once you have fixed the records, you can reload the file.

Specialized dump and load techniques

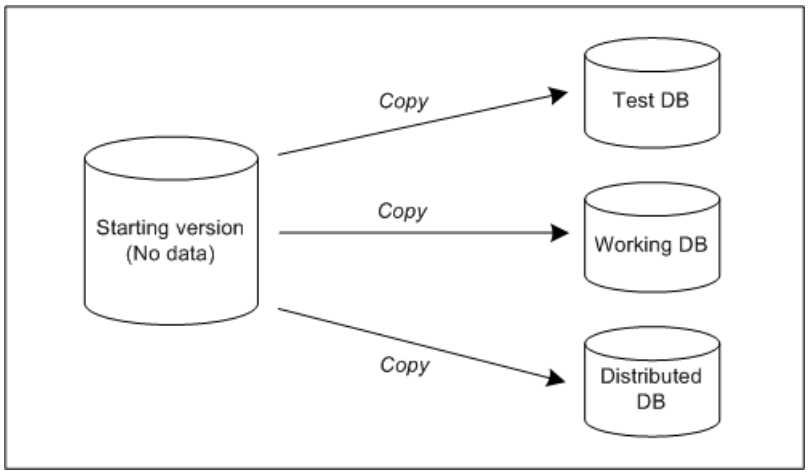
The following sections provide step-by-step instructions for dumping and loading data for specialized purposes.

Creating a starting version of a database

Suppose you finish writing an application and your database contains not only the definitions users require to start running the application, but also the data you used to test the procedures. Before you put the application to use, you want to create a version of the database that contains only data definitions and no data. This database becomes a template you can copy for testing purposes, for your own use, and for distribution.

The following figure illustrates how the OpenEdge Database Utility (PRODB) copies the starting version of a database.

Figure 55: Copying the starting version of a database



To create a new database that contains table definitions from another database, use the Dump facility to dump the table definitions, the PRODB utility to create the new database, and the Load facility to load the table definitions into the new database.

Use the following steps to create a starting version of a database:

1. Start an OpenEdge session with the database (the source) you want to copy.
2. Dump the database table definitions into a data definitions file.
3. Dump the database table data into a contents file.
4. Create a new copy of the empty database.
5. Start an OpenEdge session with the new database (the target) you created in Step 4.
6. With the data definitions file you created in Step 2, load the database definitions into the new database.

Using a constant table in multiple databases

You can use the Dump and Load utility to copy an existing table from one database into another. For example, instead of writing a procedure to input state data into the state table in a new database, you can dump the table contents from the state table into a copy of the sports database, and then load those records into the state table in another database.

To copy an existing table from one database into another:

1. Start an OpenEdge session with the database (the source) that has the table you want to copy.
2. Dump the contents of the table you want to copy into a contents file.
3. Switch from the current working database to the target database.
4. With the target database as the working database, load the contents of the table.

Economizing disk space

A database can inefficiently occupy system disk space over time, especially during development phases when frequent deletions of database elements occur. One benefit of dumping and reloading a database is better allocation of disk space.

Dumping and reloading a database involves essentially creating a new starting version of the database and loading the table contents into this new database. During the loading stage PROUTIL repartitions the disk, removing any gaps created when database elements were deleted, thereby more efficiently allocating disk space. Use PROUTIL with the TABANALYS qualifier to see how fragmented a database is. For information, see the description of PROUTIL TABANALYS in PROUTIL Utility on page 629.

To economize disk space by dumping and reloading a database:

1. Start an OpenEdge session with the database (the source) you want to copy.
2. Dump the database table definitions into a data definitions file.
3. Dump the database table data into a contents file.
4. Dump the sequence values.
5. Create a new copy of the database.
6. Designate the new (target) database you created in Step 5 as the working database.
7. With the data definitions file you created in Step 2, load the database definitions into the new database.
8. Load the table contents files you created in Step 3 into the new database, using either the Load or Bulk Loader utility.
9. Load the sequence values.
10. Delete the old database using the PRODEL utility.

Optimizing data for sequential access

If users repeatedly request data from the same tables in a determined order (for example, to create a certain report), dump and reload the data to optimize it for sequential access. Loading tables in order of average record size results in the most efficient organization of records for sequential access.

To dump and reload data to optimize it for sequential access:

1. Use PROUTIL with the TABANALYS qualifier to determine the mean record size of each table in the database. See PROUTIL Utility on page 629 for a detailed description of the PROUTIL TABANALYS qualifier.
2. Dump the definitions and data.

   The Data Administration and Data Dictionary tools dump in order of the primary index. If you access the records by another index, dump the data by that index. Use an ABL procedure similar to the following to dump the data by the index of your choice:

   ```
   OUTPUT TO table-name.d.
   FOR EACH table-name USE-INDEX index:
     EXPORT table-name.
   END.
   ```

   This procedure creates a contents (`.d`) file organized by order of access.

3. Load the tables of less than 1,000 bytes first, in order of average record size. Use the Data Dictionary or the Data Administration tool to load data one table at a time, or use the Bulk Loader utility and a description file to control the order.

   If you use the Bulk Loader, the order of the fields in the description file must match the order of the fields in the data file. If they do not match, the Bulk Loader attempts to load data into the wrong fields.

4. Load the remaining, larger tables, in order of average record size.

Optimizing data for random access

If your application accesses data in a random order, for example an online transaction-processing application, you can dump and reload the data to optimize it for random access. Use a database with small, fixed-length extents spread across the volumes to balance the I/O across the disks. Use several clients to load data simultaneously, thereby spreading the data across all disks.

This technique also improves load performance on SMP platforms.

To dump and reload data to optimize it for random access:

1. Dump the data and definitions.
2. Load the definitions using the Data Dictionary or the Data Administration tool.
3. In multi-user mode, start a server with a before-image writer (BIW) and asynchronous page writer (APW).
4. Start a client for each processor on your system. Have each client load certain tables.

   For each client, write an ABL procedure similar to the following:

```
/*Run this procedure after connecting to a database*/
DEFINE VARIABLE iNextStop AS INTEGER NO-UNDO.
DEFINE VARIABLE iRecCount AS INTEGER NO-UNDO.
DEFINE VARIABLE ix        AS INTEGER NO-UNDO.
INPUT FROM tablename.d.
TOP:
REPEAT TRANSACTION:
  iNextStop = iRecCount + 100.
  REPEAT FOR table-name WHILE iRecCount LT iNextStop
    ON ERROR UNDO, NEXT ON ENDKEY UNDO, LEAVE TOP:
    CREATE table-name.
    IMPORT table-name.
    iRecCount = iRecCount + 1.
  END.
END.
```

   The clients, loading the data simultaneously, distribute the data across all disks. This eliminates hot spots (that is, areas where data might be concentrated).

5. After the data is loaded, perform a full index rebuild using PROUTIL IDXBUILD.

Using no-integrity mode bulk data loading

PROUTIL can perform database updates faster when it runs in no-integrity mode rather than in the default full-integrity mode. However, if there is a system crash during this update activity, the database can only be recovered by restoring a backup.

To enable no-integrity mode, use the No Crash Protection (`-i`) parameter. This parameter is particularly useful for running batch jobs, such as bulk data loading. For more information about No Crash Protection, see Database Startup Parameters on page 473.

Caution: Back up databases before you run any process with the `-i` startup parameter. Any crash that occurs while you are running processes with the `-i` parameter damages the database and makes that database invalid. If this happens, you must restore the database from the most recent backup copy.

Saving disk space by deactivating indexes

You can free up disk space for an application by deactivating its indexes. Also, if you are dumping and loading a large database, you can speed up the operation by deactivating the indexes in one or more database tables. This technique should be used only in situations where indexes are used infrequently, such as in an application that generates reports only once or twice a year. There are two ways to deactivate indexes:

- Deactivate individual indexes or all the indexes with the Data Administration tool if you are using a graphical interface, or the Data Dictionary if you are using a character interface.
- Deactivate indexes from an ABL procedure.

You can deactivate (but not reactivate) a single index from either Data tool. If you create a new unique index on an existing file, consider deactivating the index. If existing table data yields duplicate keys, all changes made during the current session are backed out when you try to save them. Create or reactivate a unique index after you have ensured that all key values are unique.

To activate an index, use PROUTIL IDXBUILD. For more information about activating indexes with PROUTIL IDXBUILD, see PROUTIL IDXBUILD qualifier on page 715.

Once an index is deactivated, you cannot use it to retrieve information from the records it normally points to. If you attempt to use a `FOR`, `FOR EACH`, or `CAN-FIND` statement in connection with a deactivated index, the database engine terminates the currently executing program. However, you can create, delete, or update a record that a deactivated index points to.

The database engine does not examine the active status of an index when it makes a search. Nor does the active status of an index affect the time stamp on the `_Index` file. As a result, precompiled programs do not require compilation if the only change to the schema is index activation or deactivation.

Deactivating indexes from Data Administration

To deactivate an individual index or all of the indexes using Data Administration:

1. Access the **Data Administration**.
2. Choose **Utilities** > **Index Deactivation**. The **Index Deactivation** dialog box appears.
3. Choose **OK**. The tables in the database are listed.
4. Type `all` to deactivate all indexes, or select the indexes you want to deactivate. The Data tool prompts you to verify that you want to deactivate all the indexes.
5. Verify that you want to deactivate the specified indexes.

Deactivating indexes from the Data Dictionary

To deactivate an individual index from a graphical user interface:

1. Access the **Data Dictionary** tool.
2. Click the **Index** icon.
3. Click the index you want to deactivate.
4. Click the **Index Properties** button. The **Index Properties** dialog box appears.
5. Select the table that contains the index you want to deactivate. The Data Dictionary lists the indexes defined for the selected database.
6. Click the **Active** toggle box.

7. Click the **Save** button. The Data Dictionary prompts you to verify that you want to deactivate the index.
8. Verify that you want to deactivate the index. The Data Dictionary deactivates the index.

Deactivating indexes from an ABL procedure

You can also deactivate an index from an ABL procedure. Search through the `_Index` file to find the index you want to deactivate, then set the `_Active` field equal to `NO`. The following example uses this technique:

```
FIND _Index WHERE _Index-Name = "cust-num".
  IF (_Index._Active) THEN
     _Index._Active = FALSE.
  ELSE IF NOT(_Index._Active) THEN
    MESSAGE "The specified index is Deactivated.".
```

21

Logged Data

The OpenEdge database engine logs significant database events such as startup parameter settings; startup, shutdown, and system error messages; and application-related events. This chapter details the messages written to the database log file, and how to save key events within the database.

For details, see the following topics:

- OpenEdge database log file
- Database Log file customization
- Managing log file size
- Log file archive and truncate
- Saving key database events
- Event logging in Windows
- Client database-request statement caching

OpenEdge database log file

The OpenEdge database log is a text file that contains a history of significant database events and parameter settings. The log file includes startup parameter settings, the date and time of startup and shutdown, system error messages, and utility and maintenance activity. This file has a `.lg` extension. Entries in the database log can help you trace events preceding database crashes.

The format of each entry in the database log file provides detailed information about the time the event occurred, the source of the event, and the text of the associated message and message number. A database log entry is formatted as follows:

```
[yy/mm/dd@hh:mm:ss.uuushhmm] P-nnnnnn T-nnnnnn S name nnn: (nnnnn) Message
```

The table below describes the fields of each entry in the log file. All fields, with the exception of the message text, have a fixed width.

Table 57: Database event log format

Field in log entry	Description
yy/mm/dd	Date in year/month/day format.
hh:mm:ss.uuu	Local time in hours, minutes, seconds, milliseconds.
shhmm	Time zone relative to GMT where `s` is a `+` or `-` and `hhmm` are hours and minutes. For example, a value of -0500 represents a time zone five hours behind GMT.
P-nnnnnn	The process id of the process that produced the message. The process id is blank for remote clients. Space is allocated for a maximum of 10 digits in the process id.
T-nnnnn	The thread id of the thread that produced the message. In Windows, thread ids are unique across a system; there is no implied relationship between a thread id and the thread's creation order. On UNIX, thread ids are unique to a process. The id of the main thread is 1, and additional threads will be numbered in order.
S	The severity level of the message. There are three possible values: • **I** —Informational message • **W** — Warning message • **F** — Fatal message
name nnn:	User name and user number. These two items create single field to identify the source of the message. Possible values include: BROKER 0:, USR 5:, RPLS 5:, DBUTIL:, etc. Nine spaces are allocated for this field. The name portion is left justified and the user number is right justified.
(nnnnn)	The message number of the message. More details about the message can be obtained by accessing **Help** > **Messages...** from the OpenEdge Desktop.
Message text...	The text of the message.

Database Log file customization

New startup parameters give you the opportunity to tailor messages written to the log file. You can limit payload information for security, and you can omit certain messages from being written at all. The parameters are:

- **Limit log file payload (`-limitLgPayload`)** — Prevent message payload from being written to the database log file. Message *payload* is defined as the user name, tty name, IP address, and connection hostname in a message. For details, see Limit log file payload (-limitLgPayload) .
- **Omit log messages (`-omitLgMsgs`)** — Specify up to 16 non-fatal messages you do not want written to the database log file. For details, see Omit Log Messages (-omitLgMsgs).

Managing log file size

The database log (`lg`) file expands as you use the database. If it becomes too large, you can reduce its size by removing old log entries. To remove log entries from an `lg` file, use the Log Maintenance (PROLOG) utility or a text editor. Preserve the information in your log file by backing it up prior to truncation.

Enter the following command to remove entries from an event log file:

```
prolog database-name [ -online ]
```

The PROLOG utility removes all but the most recent entries from the log file. Use the `-online` parameter when you need to truncate the log file for a database that is online. For complete details on the PROLOG utility, see PROLOG utility on page 843.

Log file archive and truncate

Database log files grow constantly. The ability to truncate a database log file, and optionally archive the file before truncation while the database is online, provides DBAs with the ability to save log file information, and also manage file growth.

As you configure your database to truncate and possibly archive your log file, consider the following:

- There is only one active log file, and it is named *dbname*`.lg`.
- When the log file is truncated, licensing information and the startup parameters of the primary broker are re-written to the log file.
- If truncation is configured to occur at a specific time, and the database is offline or in single-user mode, the log file is not truncated.
- If truncation is configured to occur when the log file reaches a specific size, and the database is offline or in single-user mode when the size is reached, the log file is not truncated. If the truncation size parameters are met the next time the database is brought online in multi-user mode, the log file is truncated then.
- If the database is in the middle of writing a message when the truncation size or time is reached, the log file is truncated after the message writing completes.
- You can configure your log file to be truncated by both time and size, and both are honored.

- If you choose to archive your log file, in the time between starting your database and archiving the log file, an interim log file is created and maintained in the archive directory or the database directory if no archive directory is specified.

You can manage the criteria for truncating and archiving your log file through startup parameters, the _DbParams VST, or with PROMON.

Log file truncate and archive startup parameters

Use the following startup parameters to configure truncating and archiving your database log file.

Table 58: Log file truncation parameters

Parameter	Description	For more information, see
`-lgTruncateFrequency`	Specify the frequency of when your log file is truncated. Valid values include daily, one day per week, or the last day of the month. Must be used with `-lgTruncateTime`.	Log file truncate frequency (-lgTruncateFrequency)
`-lgTruncateTime`	Specify the time of day when your log file is truncated. Valid values are based on a twenty four hour clock, 00:00 (midnight) to 23:59 (one minute before midnight). Must be used with `-lgTruncateFrequency`. **Note:** If the database is offline or in single user mode when the time and frequency parameters indicate, no truncation occurs until the next time the parameters are satisfied.	Log file truncate time (-lgTruncateTime)
`-lgTruncateSize`	Specify the size in MB when the database log file is to be truncated. Valid values range from 0 to 2147483647. The value of 0 (default) indicates that truncation based on size is disabled. **Note:** If the database is in single user mode when the log file reaches the specified size, it is not truncated. It is truncated the next time the database is brought online in multi-user mode, provided the size criterion is still met. If the log file reaches the specified size while writing a message, the database log file is truncated at the conclusion of writing the message.	Log file truncate size (-lgTruncateSize)

The table that follows describes the parameters for archiving your database log file. If you do not configure truncation of your database log file, it is not archived.

Table 59: Log file truncation parameters

Parameter	Description	For more information, see
-lgArchiveEnable	Specify that you want the database log file archived before truncating it. **Note:** If archiving the log file fails for any reason, the log file is not truncated.	Log file archive enable (-lgArchiveEnable)
-lgArchiveDir	Specify the fully qualified path name of the directory where you want the archive files written.	Log file archive directory (-lgArchiveDir)

PROMON Database Log file management

PROMON provides the ability to view and change the current settings for truncating and archiving log files. To access the **Database Log file management** from the PROMON main menu, select **R&D (Advanced options)** > **4 (Administrative Functions)** > **15 (Database Log file management)**. The following menu appears:

```
02/05/18        OpenEdge Release 11 Monitor (R&D)
15:11:14        Database log file management

        Current .lg file maintenance settings:

          1. Omit .lg file messages (-omitLgMsgs):             Not Enabled
          2. Limit .lg file payload (-limitLgPayload):         Not Enabled
          3. Archive directory (-lgArchiveDir):                Not Enabled
          4. Archive before truncation (-lgArchiveEnable):     Not Enabled
          5. Truncation frequency (-lgTruncateFrequency):      Not Enabled
          6. Maximum size before truncation (-lgTruncateSize): Not Enabled
          7. Time of day for truncation (-lgTruncateTime):     00:00


Enter a number, P, T, or X (? for help):
```

For each option, select the number to toggle the setting, or input a new value.

You can also determine the last time the log file was truncated (since startup). From the PROMON main menu, select **7 (Database Status)**.

_DbParams VST

All the startup parameters for truncating and archiving your database log file are modifiable while your database is online using the _DbParams VST.

Archiving

When you enable archiving of your truncated log file, the following process occurs:

1. An interim archive file is created in your archive directory, or in the database directory, if an archive directory is not specified. The interim archive file is named,*dbname*.lg.*date-time*.*n*.tmplg, where the *date-time* is the time the temporary archive is created.
2. Over time, the database broker writes log file entries to the interim archive in 8K chunks.
3. When the database log file is truncated, the broker writes any remaining log messages to the interim archive.
4. The interim archive is re-named to the archive file name, using the *date-time* when the truncation and archive criteria are met. The file is named, *dbname*.lg.*date-time*.*n*.
5. A new temporary archive is created the next time the broker has an 8K chunk to write to it.
6. Repeat Steps 2-5.

The temporary archive is deleted at database shutdown, or if the database log file is truncated by PROLOG.

Archive file naming convention

When you archive your log file, it is given a name using the convention, *dbname*.lg.*date-time*.*n*.

In this file name:

- *dbname* — The name of the database.
- *date-time* — The date and time when the archive file is created, using the format YYYY-MM-DD.HH-MM.
- *n* — An identifier, only incremented when the combination of *dbname* and *date-time* does not create a unique file name.

The interim archive file uses the same convention, but adds a .tmplg extension to the file name.

Truncated and archived files

As you truncate and archive your log files, the log and archive files are modified as follows:

- **Truncated log file** — When you truncate your log file, all existing information in your log file is deleted, and the log file is re-created. Near the top of your log file is the following message:

  ```
  (19219) The database log file was truncated.
  ```

  Your log file is also re-loaded with all your license information, and the primary broker startup parameters with their current values (which could have been changed while the database is online).

- **Interim archive file** — When you specify that you want to archive your log file, an interim archive is created and filled, in anticipation of being archived. When you enable archiving, the interim log file is named,*dbname*.lg.*date-time*.*n*.tmplg, where the *date-time* is initially the start time, and over time is the time of the previous archive. The interim log file is renamed when archiving occurs, and a new interim log file is created. The interim log file is deleted when the database is shut down or the log file is truncated with PROLOG.

- **Truncated and archived log file** — When you truncate and archive your log file, the log file is first archived, then all existing information in your log file is deleted, and the log file is re-created. The following messages are written to your truncated log file:

  ```
  The database log file was archived to archive-dir/archive-file.
  (19219) The database log file was truncated.
  ```

In this instance, *archive-dir* is the fully qualified directory name where the log file is created, and *archive-file* is the name of the log file archive.

Your log file is also re-loaded with all your license information, and the broker startup parameters with their current values (which could have been changed while the database is online).

- **Archived log file** — When your archive log file is created, near the end of the archive, are the following lines (written to the log file when preparing to archive):

```
(19233) Preparing to truncate the database .lg file.
(19232) Database .lg file archive in process.
```

Saving key database events

Saving key events can provide a database administrator or a Progress technical support engineer an efficient record of critical events in the history of your database. The following section defines key events and their processing. Topics discussed are:

- Defining key events on page 445
- Saving key events on page 446
- Enabling save key events on page 446
- Disabling save key events on page 447
- Stored key events on page 447
- _KeyEvt table on page 452

Defining key events

Key events in the history of your database include starting the database, changing the structure of the database, executing certain utilities, and encountering errors. When a key event occurs, details of the event are written to your database log file. When you enable your database to save key events, the information written to the log file is also copied into the `_KeyEvt` table in the database. The details of key events provide insight into critical moments in the history of your database.

Having the details of key events preserved is useful in the diagnosis of issues, either independently, or with the assistance of Progress technical support. Saving key events in the `_KeyEvt` table ensures that vital information is not lost when your log file is truncated or deleted. It also provides quick access to information that is otherwise embedded within your log file, possibly in the middle of millions of log file entries.

Saving key events

When you enable your database to save key events, an additional database process is created to handle the responsibility. This background database process periodically scans your database log file, creating records of the key events in the _KeyEvt table. When successfully started, the key event process writes messages to your log file similar to the following:

```
[2006/04/06@10:52:58.578-0400] P-6450       T-1     I KEVTRDR 5: (13658)
Starting the Save Key Events daemon.
[2006/04/06@10:52:58.578-0400] P-6450       T-1     I KEVTRDR 5: (2518)
Started.
```

The key event process is also visible in PROMON. The following example shows **User Control** output from a database enabled to save key events:

```
User Control:
Usr Name     Type Wait      Area    Dbkey     Trans     PID  Sem Srv   Login  Time
  0 docdemo  BROK  --          0        0         0  11117   0   0 04/07/06 13:50
  5 docdemo  MON   --          0        0         0  11220   2   0 04/07/06 13:50
  6          KERD  --          0        0         0  11235   3   0 04/07/06 13:50
RETURN - repeat, U - continue uninterrupted, Q - quit:
```

The new process type, KERD, represents the key event process.

Enabling save key events

Enabling your database to save key events can be done at any time. The first time a broker is started for the database after key events are enabled, **all** existing key events in your log file will be recorded into your _KeyEvt table.

To enable your database to save key events:

1. Optionally define a database area for your _KeyEvt table and add it to your database. If you don't specify an area, the table is created in your Schema Area.
2. Enable your database to save key events with the PROUTIL ENABLEKEYEVENTS utility. See the PROUTIL ENABLEKEYEVENTS qualifier on page 1205 for details on the utility.

The following example demonstrates these steps:

```
# show the contents of add_keyevt.st
$ cat add_keyevt.st
  #
  # Add area for key event storage
  #
  d "Key Event Area":26,32;8 .
  #

$ prostrct add mydb add_keyevt.st
$
$ proutil mydb -C enablekeyevents "Key Event Area"
OpenEdge Release 10.1A01 as of Thu Apr  6 20:40:49 EDT 2006

Key Events has been enabled for database mydb. (12479)
$
```

Note: If you add a new area to your database for key events, update your structure description file with PROSTRCT LIST. For more information on PROSTRCT, see PROSTRCT Utility on page 785.

Disabling save key events

Disabling your database from saving key events can be done at any time, even when the database is online. Issue the following command to disable the save key event storage:

```
proutil mydb -C disablekeyevents
```

See PROUTIL DISABLEKEYEVENTS qualifier on page 686 for details on the utility.

Stored key events

The table below lists key database events and the records stored in your database when your database is enabled to save key events.

Table 60: Key events and stored records

Database activity	Key events record stored
Database startup	• Date and time of startup • Database startup parameters, one parameter per key event record • Current database high water mark
Database shutdown	Date and time of shutdown
Lock table overflow	Date and time of overflow
Protrace generation	• Date and time the protrace is generated • Name and location of the protrace file
DBTOOL	• Date and time the utility started • Data and time the utility ended
PROBKUP	• Date and time the utility started • Location of backup file • Size of backup file • Data and time the utility ended

Database activity	Key events record stored
PROCOPY	• Date and time the utility started • Source database location • Target database location • Date and time the utility ended
PROLOG	Date and time the utility started
PROMON RESOLVE LIMBO TRANSACTION	• Date and time the utility started • Transaction id • Action (commit, abort)
PROREST	• Date and time the utility started • Location of backup file • Data and time the utility ended
PROSTRCT ADD	Date and time the utility started
PROSTRCT BUILDDB	• Date and time the utility started • Name of the structure description (`.st`) file
PROSTRCT CREATE	Date and time the utility started
PROSTRCT REMOVE	• Date and time the utility started • Names and locations of the deleted extents
PROSTRCT REORDER	Date and time the utility started
PROSTRCT REPAIR	Date and time the utility started
PROSTRCT UNLOCK	Date and time the utility started
PROUTIL 2PHASE BEGIN	Date and time the utility started
PROUTIL 2PHASE COMMIT	Date and time the utility started
PROUTIL 2PHASE END	Date and time the utility ended
PROUTIL 2PHASE MODIFY	Date and time the utility started

Database activity	Key events record stored
PROUTIL 2PHASE RECOVER	Date and time the utility started
PROUTIL BIGROW	• Date and time the utility started • Number of clusters • Options specified (`-r`) • Date and time the utility ended
PROUTIL BULKLOAD	• Date and time the utility started • Date and time the utility ended
PROUTIL CONVCHAR (`convert` option only)	• Date and time the utility started • Codepage specified
PROUTIL CONVFILE (`convert using` option only)	• Date and time the utility started • Name and location of the file being converted
PROUTIL DBAUTHKEY	Date and time the utility started
PROUTIL DISABLEAUDITING	Date and time the utility started
PROUTIL DISABLEKEYEVENTS	Date and time the utility started
PROUTIL DISABLEJTA	Date and time the utility started
PROUTIL DUMP	• Date and time the utility started • Name of table being dumped • Target directory of the dump • Date and time the utility ended
PROUTIL DUMPSPECIFIED	• Date and time the utility started • Name of table being dumped • Name of index field used for selecting records • Target directory of the dump • Date and time the utility ended

Database activity	Key events record stored
PROUTIL ENABLEAUDITING	Date and time the utility started
PROUTIL ENABLEKEYEVENTS	Date and time the utility started
PROUTIL ENABLEJTA	Date and time the utility started
PROUTIL ENABLEPDR	Date and time the utility started
PROUTIL IDXACTIVATE	Date and time the utility started
PROUTIL IDXBUILD	• Date and time the utility started • Options specified (`-TB`, `-TM`, `-SG`) • Date and time the utility ended
PROUTIL IDXCHECK	Date and time the utility started
PROUTIL IDXCOMPACT	• Date and time the utility started • Table name and index name of the index being compacted • Degree of compaction • Date and time the utility ended
PROUTIL IDXFIX	• Date and time the utility started • Date and time the utility ended
PROUTIL IDXMOVE	• Date and time the utility started • Table and index name of the index being moved • Target area name where the index is being moved • Date and time the utility ended
PROUTIL LOAD	• Date and time the utility started • Name of file being loaded • Date and time the utility ended
PROUTIL MVSCH	Date and time the utility started.

Database activity	Key events record stored
PROUTIL TABLEMOVE	• Date and time the utility started • Name of the table being moved • Target area name where the table is being moved • Target area where indexes with on the table are optionally moved • Date and time the utility ended
PROUTIL TRUNCATE AREA	• Date and time the utility started • Date and time the utility ended
PROUTIL TRUNCATE BI	• Date and time the utility started • Option specified (`-bi`) • Date and time the utility ended
PROUTIL TRUNCATE BI -F (Force access to damaged database)	Date and time the utility started
PROUTIL UPDATEVST	Date and time the utility started
RFUTIL AIMAGE BEGIN	Date and time the utility started
RFUTIL AIMAGE END	Date and time the utility started
RFUTIL AIMAGE AIOFF	Date and time the utility started
RFUTIL AIMAGE EXTENT EMPTY	• Date and time the utility started • Extent number or pathname of extent to empty
RFUTIL AIMAGE EXTRACT	• Date and time the utility started • Name of the after-image extent having its notes extracted • Date and time the utility ended
RFUTIL AIMAGE NEW	Date and time the utility started

Database activity	Key events record stored
RFUTIL AIMAGE TRUNCATE	• Date and time the utility started • Specified block size (`-aiblocksize`) • Date and time the utility ended
RFUTIL MARK BACKEDUP	Date and time the utility started
RFUTIL ROLL FORWARD	• Date and time the utility started • Name of the after-image extent to roll forward • Specified option (`-r`) • Date and time the utility ended
RFUTIL ROLL FORWARD RETRY	• Date and time the utility started • Name of the after-image extent to roll forward • Specified option (`-r`) • Date and time the utility ended

_KeyEvt table

The `_KeyEvt` table contains records generated by the key event process. While the `_KeyEvt` table is considered part of the database metaschema, it has properties that differentiate it from other metaschema tables and the schema is detailed, as discussed in the following sections:

- _KeyEvt table creation on page 452
- _KeyEvt schema on page 453
- _KeyEvt table maintenance on page 453

_KeyEvt table creation

The `_KeyEvt` table is created when your database is first enabled to save key events. By default, the `_KeyEvt` table is created in the Schema Area, but you can specify a different area.

The following example demonstrates how to enable key events for a database named `my-db`, and place the `_KeyEvt` table in the Schema Area:

```
$ proutil my-db -C enablekeyevents
```

_KeyEvt schema

The _KeyEvt table stores key events in the database. The table below describes the active fields of the _KeyEvt table.

Table 61: _KeyEvt table schema

Field name	Description	Data type	Format
_KeyEvt-Date	The date and time, in DATETIME-TZ format, when the event occurred	DATETIME-TZ	99/99/9999 HH:MM:SS.SSS+HH:M
_KeyEvt-Procid	The process id of the process that logged the key event message	INTEGER	->,>>>,>>>,>>9
_KeyEvt-ThreadId	The thread of the process that logged the key event message	INTEGER	->,>>>,>>>,>>9
_KeyEvt-MsgId	The message number of the event	INTEGER	->,>>>,>>>,>>9
_KeyEvt-Usrtyp	The type of user that logged the event, for example, BROKER, DBUTIL, KEVTRDR, Usr, etc.	CHARACTER	X(32)
_KeyEvt-Usrnum	The user number of the user that logged the event	INTEGER	->,>>>,>>>,>>9
_KeyEvt-Event	The message text associated with the key event	CHARACTER	X(200)

_KeyEvt table maintenance

Maintenance on the _KeyEvt table treats the table like a standard table. The following actions are allowed:

- **Table move** — Move the _KeyEvt table to a new area with PROUTIL TABLEMOVE.
- **Truncate area** — Delete the data in the _KeyEvt tables and indexes with PROUTIL TRUNCATE AREA.
- **Index move** — Move a _KeyEvt table index with PROUTIL IDXMOVE.
- **Record deletion** — Records in the _KeyEvt table can be deleted with standard 4GL and SQL statements.

Event logging in Windows

In addition to the OpenEdge database log, the OpenEdge Server writes events to the Windows Event Log. The Event Log is the object that enables Windows users to view the status of application, security, and system processes, and to view their associated events. The OpenEdge database is an application process that writes events to the Application Event Log. You use the Event Viewer to see the Event Log's contents. You can customize the Event Viewer so that it displays only the event types that you want to view. You access the Event Viewer through the Administrative Tools program group. For a list of Windows operating systems currently supported by OpenEdge, see *OpenEdge Getting Started: Installation and Configuration*.

The components that enable OpenEdge services to log messages to the Application event log database are described in the table below.

Table 62: Event logging components

This component . . .	Performs this function . . .
The Event Viewer	Enables users to view the Event Log.
The Event Log	Records event information.
`PROMSGS.DLL`	Contains database messages for the event log.
`CATEGORY.DLL`	Contains the 14 categories into which database messages might fall.
`PROMSGS file`	Contains the full set of the database messages including translated versions. The PROMSGS file is installed at the top level of the OpenEdge install directory with the executable.

Managing OpenEdge RDBMS events in Windows

You can define the level of event logging that you want your OpenEdge application to support by using the Event Level Environment Variable (`EVTLEVEL`), or the Event Level startup parameter (`-evtlevel`).

The table below lists valid event level values.

Table 63: Event level values

Value	Description
None	No events are written to the event log.
Brief	Messages defined as Error and Warning messages are written to the event log.
Normal	Messages defined as Error and Warning messages are written to the event log. In addition, any message that is normally written to the log file (`.lg`) is also written to the Event Log. This is the default.
Full	Every message generated is written to the Event Log. Any messages generated using the Message Statement are also written to the log file.

Understanding the event log components

The components of the Event Log are standards defined by Windows. The table below describes the Event Log columns.

Table 64: Event Log components

Column	Description
Date	Date the event occurred.
Time	Time the event occurred.

Column	Description
Source	Source of the event. If the source of the event is a connected OpenEdge database, the database's name appears in this column.
Category	There are 16 OpenEdge event categories: AIW, APPBROKER, APPSERVER, APW, BACKUP, BIW, DATASERVER, MON, OIBRKR, OIDRVR, OpenEdge, RFUTIL, SERVER, SHUT, USER, and WDOG. The APPBROKER and APPSERVER categories appear in the Event Log only when messages are logged by the AppServer™ application broker and application server, respectively.All categories reside in a file called `category.dll`. These categories correspond to the existing categories of events that are displayed in the `progress.lg` file (application broker and application server events are displayed in the AppServer log file, `proapsv.lg`).
Event	The OpenEdge message number that was generated. These are the same message numbers that are displayed in the standard `database .lg` file.
User	Identifies the user logged in to the Windows workstation where the event occurred.
Computer	Identifies the name of the Windows workstation where the event occurred. The Event Viewer enables you to get more information about events by double-clicking on any event.

Select an event to view additional information about it.

The Event Log and the registry

Windows requires that applications using the Event Log be bound to all of the necessary components. This means that the `PROMSGS.DLL` and the `CATEGORY.DLL` must be bound to an OpenEdge database. The database engine stores this information in the registry. The engine makes the registry entries and performs any binding operations that are necessary when you initially access a database. When the engine binds the `.dll` files to the database, it writes the fully qualified pathname to the registry. If you delete the database, you must manually remove the associated data from the registry. Or, if you move the location of the `.dll`s after you access the database, you must manually edit the registry data. The OpenEdge components can be found in the following location in the registry:

```
HKEY_LOCAL_MACHINE
SYSTEM
 CurrentControlSet
  Services
   EventLog
    Application
     PROGRESS
     <Database Name>
```

See the Microsoft Windows documentation for more information about editing registry files.

When the database engine tries to find the `.dll`s before this information is included in the registry, it searches the current directory. If the `.dll` is not in the current directory, the engine searches the directory where the executable is located. If the `.dll` is not in the same directory as the OpenEdge executable, the engine searches the user's path. If the `.dll` is not in the user's path, the engine generates a message stating that the `.dll` cannot be found, and it writes a message to the OpenEdge event log file.

Client database-request statement caching

OpenEdge supports database request statement caching. This functionality enables the database administrator to monitor ABL and SQL driven database activity using PROMON or the _Connect Virtual System Table (VST). Using database request statement caching, the database administrator can track client activity and identify line numbers within an ABL program (or an SQL statement) associated with database requests.

Note: For information on configuring client database-request statement caching, see PROMON utility on page 544.

The database-request statement cache is updated by individual clients, and is refreshed each time a client performs a database operation. The cache can display ABL or SQL activity, including:

- ABL program names and line numbers of ABL code executing database requests
- ABL program stacks program names and line numbers, beginning with the current program name and line number executing a database operation. You can access the entire call stack from PROMON; the _Connect VST holds up to 32 entries.
- Recent SQL statements

Client database-request statement caching for SQL provides the SQL statements that generate database server requests; for ABL, client database-request statement caching provides program names and line numbers of the ABL code that generated database server requests.

Client database-request statement caching can be used to identify statements that are potential performance problems. For example, multi-database queries are often complicated by needing to transfer large amounts of data typically produce complex result sets. If processing large amounts of data results in poor query response time, you might want to monitor use statement caching to monitor the ABL or SQL database activity and identify areas within an ABL program (or SQL statements) associated with the database requests as code to consider re-writing.

Note: Client database-request statement caching may negatively impact application performance due to extraneous APIs (for self-serve connections) and TCP/IP messages sent from the client to the server.

When client database-request statement caching is activated, ABL stack traces indicate which procedure executed a database operation, and what procedure generated the request. Each line in the request contains the line number and additional procedure information, such as the name of the procedure, whether it is an internal or external procedure, a user-defined function, or the method of a class.

The format of the ABL stack trace is:

```
line-number : procedure information
```

The line number indicates the point in the ABL code where the statement was executing when the ABL stack trace was generated. In some cases, the line number may be -1, indicating the end of the procedure before returning to its caller. Line number information is found in the debug listing file.

Note: The debug listing file is generated by the **DEBUG-LIST** option in the **COMPILE** statement. For more information, see *OpenEdge Getting Started: ABL Essentials*.

The procedure information contains the following information:

- The name of the procedure as it was executed by the RUN statement in the ABL application. For example, if the ABL statement used either `RUN test.p` or `RUN test.r`, then `RUN test.p` or `RUN test.r` appear as the name of the procedure.
- Class names without the `.CLS` extension.
- A temporary file with (with a `.ped` extension) is displayed as the procedure name for procedures executed using the Procedure Editor.
- Statements contained inside an internal procedure, a user-defined function, or a method of a class, precede the procedure or class name. For example, if the statement was inside an internal procedure named `myproc` in `test.p`, the procedure information contains `myproc test.p`.

IV

Reference

For details, see the following topics:

- Startup and Shutdown Commands
- Database Startup Parameters
- PROMON Utility
- PROUTIL Utility
- PROSTRCT Utility
- RFUTIL Utility
- Other Database Administration Utilities
- SQL Utilities
- Virtual System Tables

22

Startup and Shutdown Commands

This chapter describes the OpenEdge RDBMS startup and shutdown commands in alphabetical order. It describes the purpose, syntax, and primary parameters for each command.

For details, see the following topics:

- Startup command syntax
- Database startup and shutdown commands
- PROAIW command
- PROAPW command
- PROBIW command
- PROQUIET command
- PROSERVE command
- PROSHUT command
- PROWDOG command

Startup command syntax

The conventions used in command syntax descriptions is as follows:

```
command  [ db-name ] [ parameter ] [ value ] . . .
```

For example, the following command allows 100 users to access the sports database and then set values for the database connection, performance, and network parameters:

```
proserve sports -n 100 -B 30000 -L 1000 -S sprtsv -H sys27
```

The table below describes each of the command components.

Table 65: Command components

Component	Description
command	On UNIX, the command runs a script that executes an OpenEdge executable with appropriate parameters. In Windows, some commands run a batch file that executes an OpenEdge executable with appropriate parameters. Other commands run an OpenEdge executable directory.
db-name	Name of the database you want to connect to.
parameter, qualifier	Operating criteria for the command.
value	Numeric value or file specification for the parameter.

Note: On UNIX and in Windows, enter parameters exactly as shown in the syntax descriptions.

Database startup and shutdown commands

Database startup commands start OpenEdge database processes. The table below summarizes the tasks performed by each command.

Table 66: Database startup and shutdown commands

To . . .	Use . . .
Start an OpenEdge server group	`proserve -servergroup` *server-group-name*
Start a server or broker for a multi-user OpenEdge database	`proserve` *db-name* `-S` *service-name* `-H` *host-name* `-N` *network-type*
Shut down a multi-user server or broker for an OpenEdge database	`proshut` *db-name*
Start an asynchronous page writer (APW)[10]	`proapw` *db-name*
Start a before-image writer (BIW)[10]	`probiw` *db-name*
Start an after-image writer (AIW)[10]	`proaiw` *db-name*

[10] Option available only on Enterprise Database products.

To . . .	Use . . .
Stop all writes to database files by enabling a "quiet" processing point[10]	`proquiet` *db-name parameter*
Start the Watchdog utility	`prowdog` *db-name*
Shut down a remote OpenEdge DataServer	`proshut` *db-name* `-S` *service-name* `-H` *host-name* `-N` *network-type*
Shut down an APW, AIW, BIW, or Watchdog process	`proshut` *db-name* Choose option 1 (Disconnect a User) to disconnect the process.

PROAIW command

Starts the after-image writer (AIW) process.

Syntax

```
proaiw db-name
```

Parameters

db-name

Specifies the database where you want to start the AIW process.

The AIW improves performance by writing notes to the after-imaging file. For more information on the AIW, see Managing Performance on page 275.

Notes

- To stop the AIW, disconnect it by using the PROSHUT command. You can start and stop the AIW at any time without shutting down the database.
- The AIW counts as one user. You might have to increment the value of the Number of Users (`-n`) parameter to allow for the AIW. However, the AIW does not count as a licensed user.
- You can increase the number of buffers in the after-image buffer pool by using the After-image Buffers (`-aibufs`) parameter. Increasing the number of buffers when running an AIW increases the availability of empty buffers to client and server processes. Increasing the After-image Buffers parameter has no effect if the AIW is not running.
- After-image writers are only available on Enterprise systems.

PROAPW command

Starts an asynchronous page writer (APW).

Syntax

```
proapw db-name
```

Parameters

db-name

Specifies the database where you want to start an APW.

APWs improve database performance by performing overhead operations in the background. For more information on APWs, see Managing Performance on page 275.

Notes

- To stop an APW, disconnect it by using the PROSHUT command. You can start and stop APWs at any time without shutting down the database.
- Each APW counts as a user. You might have to increase the value of the Number of Users (`-n`) parameter to allow for APWs. However, APWs do not count as licensed users.
- The optimal number depends on your application and environment. To start, use one page writer for each disk where the database resides. If data gathered from PROMON indicates that this is insufficient, add more. For more information on PROMON, see PROMON Utility on page 543.
- For an application that performs many updates, start one APW for each disk containing your database, plus one additional APW. Applications that perform fewer changes to a database require fewer APWs.
- Asynchronous page writers are only available on Enterprise systems.

PROBIW command

Starts a before-image writer (BIW) process.

Syntax

```
probiw db-name
```

Parameters

db-name

Specifies the database where you want to start a BIW.

The BIW improves database performance by performing before-image overhead operations in the background. For more information on the BIW, see Managing Performance on page 275.

Notes

- To stop the BIW process, disconnect it by using the PROSHUT command. You can start and stop the BIW at any time without shutting down the database.

- The BIW process counts as one user. You might have to increment the value of the Number of Users (`-n`) parameter to allow for the BIW. However, the BIW does not count as a licensed user.
- You can increase the number of before-image buffers with the Before-image Buffers (`-bibufs`) parameter. Increasing the number of buffers increases the availability of empty buffers to client and server processes.
- Before-image writers are only available on Enterprise systems.

PROQUIET command

Stops all writes to database files by enabling a "quiet" processing point.

Syntax

```
proquiet dbname -C { { enable [ nolock ] | disable [ markbackedup ] } | bithreshold
 n }
```

Parameters

db-name

Specifies the name of the database where you are enabling or disabling a quiet processing point.

`enable | disable`

Enables or disables a quiet processing point. Any processes that attempt transactions while a quiet point is enabled must wait until the quiet point is disabled.

`nolock`

Allows you to enable a quiet point without waiting on shared memory latches.

`markbackedup`

Allows you to mark the database as backed up as the quiet point ends.

`bithreshold` *n*

Specifies the maximum size to which BI recovery files can grow, where *n* is an integer specifying the size of the threshold in MB. You can increase the size of the threshold above the current value or reduce the size to one cluster larger than the size of the recovery log file at the time the PROQUIET command is issued.

Note: Though the above table lists the `-C` parameter to show the complete syntax, you do not need to use the `-C` parameter in the PROQUIET syntax.

PROQUIET ENABLE stops all writes to the database; PROQUIET DISABLE ends the quiet point to resume writes. PROQUIET is useful for advanced backup strategies. You can also use the PROQUIET command with the `bithreshold` parameter to adjust the size of the recovery log threshold online. Use the PROSERVE command with the `-bithold` startup parameter to set the size of the primary recovery log threshold on startup.

For more information on using database quiet points, see Backing Up a Database on page 97 and Managing Performance on page 275.

Notes

- Enabling a quiet point on a database with after-imaging enabled forces an AI extent switch.
- Enabling a no-lock quiet point on a on a database with after-imaging enabled does **not** force an AI extent switch. The BUSY AI extent at the time the no-lock quiet point is enabled must be rolled forward with ROLL FORWARD RETRY. For more information, see RFUTIL ROLL FORWARD RETRY qualifier on page 817
- Disabling a quiet point with the `markbackedup` option marks the database in the same manner as the RFUTIL MARK BACKEDUP command, but allows the database to stay online. For more information on the RFUTIL command, see RFUTIL MARK BACKEDUP qualifier on page 815.
- Quiet points are only available on Enterprise systems.

Example

- Use the PROQUIET command to manage the primary recovery area (BI) threshold before your database stalls.

  For example, to start a server and set a 500MB BI threshold, allowing the system to stall if that threshold is reached, use the PROSERVE command as follows:

```
proserve mydemo -bithold 500 -bistall
```

  Assume a long running transaction causes the expansion of the BI, but before the threshold is reached you receive the following message:

```
BI file size has grown to within 90% of the threshold value 523763712.
(6559)
```

  After receiving message 6559, you decide to increase the BI threshold to 1GB while the database remains online and investigate the cause of the unexpected BI growth before the system stalls. Use the PROQUIET command, as follows:

```
proquiet mydemo bithreshold 1000
```

  The above command establishes a quiet point and increase the threshold. The database does not stall.

  Note: In practice, invoke PROQUIET commands by using a script so that they occur as quickly as possible and with the least amount of impact to the online system.

- When a database stalls because the BI threshold is reached, the stall causes an implicit quiet point and the database engine writes a message to the log file. To expand the BI threshold and continue forward processing, use the PROQUIET command with the `bithreshold` parameter only, as follows:

```
proquiet mydemo bithreshold 1000
```

PROSERVE command

Starts the broker, which in turn spawns the server. The server process coordinates all access to the specified OpenEdge database.

Syntax

```
proserve { db-name | -servergroup [ server-group-name ] [ parameters ] }
```

Parameters

db-name

Specifies the specific database you want to start.

-servergroup *server-group-name*

Specifies the logical collection of server processes to start. The *server-group-name* you specify must match the name of a server group in the conmgr.properties file. You create server groups using OpenEdge Management or OpenEdge Explorer, which are saved in the conmgr.properties file.

parameters

Specifies the startup parameters for the broker/server. See Database Startup Parameters on page 473 for a list of broker/server startup parameters.

Notes

- You can specify only one database name when using PROSERVE to start a broker or server group.
- Server groups manage network connections four separate ways:
  - Accept no network connections
  - Accept SQL and ABL network connections
  - Accept only SQL network connections
  - Accept only ABL network connections
- Typically, server groups share common attributes such as connection port, number of servers, and how connected clients are distributed among the severs.
- You create server groups using OpenEdge Management or OpenEdge Explorer, which are saved in the conmgr.properties file. The *server-group-name* you specify with the PROSERVE -servergroup parameter must match the name of a server group in the conmgr.properties file. Do not edit the conmgr.properties file directly. Instead, use OpenEdge Management or OpenEdge Explorer. For more information on OpenEdge Management and OpenEdge Explorer, see the online Help.
- The behavior of the -servergroup parameter is similar to the behavior of the -pf (parameter file) parameter. In effect, -servergroup causes a server to load the parameters associated with the server group, including the database name.

It is possible to override the parameter values associated with a server group by adding additional parameters to the PROSERVE command. For example, if the database buffer pool is set to 10,000 within the configuration associated with a server group, you can specify a larger value by adding an additional parameter:

```
proserve -servergroup sports2000.myconfig.4GLdefault -B 20000
```

Conversely, if you specify a startup parameter before the `-servergroup` parameter, the startup parameter can be overridden when the same parameter is set in the server group configuration file. For example, if you place the additional parameter before the `-servergroup` parameter, the database buffer pool remains 10,000:

```
proserve -B 20000 -servergroup sports2000.myconfig.4GLdefault
```

PROSHUT command

Shuts down the OpenEdge database server and individual processes. Before you shut down the broker, have all application users quit their sessions. If necessary, you can disconnect users by using the PROSHUT command's Disconnect a User or Unconditional Shutdown parameters.

Syntax

```
proshut db-name [ -b | -by | -bn
                | -C list | -C disconnect username
                | -F | -Gw
                | -H host-name | -S service-name
                | -shutdownTimeout [ immed | maximum | n
                    [ h | m | s ] ]
                | -cpinternal codepage | -cpstream codepage
              ] . . .
```

Parameters

db-name

Specifies the database the server is running against.

`-b`

Directs PROSHUT to perform a batch shutdown. When no client is connected, the database automatically shuts down. When one or more clients are connected, PROSHUT prompts the user to enter "yes" to perform an unconditional batch shutdown and to disconnect all active users; or "no" to perform a batch shutdown only if there are no active users. The `-b` parameter combines the functionality of the `-by` or `-bn` parameters.

-by

Directs PROSHUT to perform an unconditional batch shutdown and to disconnect all active users. Note that PROSHUT does not give users any notice before disconnecting them.

-bn

Directs PROSHUT to perform a batch shutdown only if there are no active users.

-C list

Lists all of the users connected to the database. The list is printed out to the screen without any page breaks. Use of this parameter is limited to local non-networked connections only.

-C disconnect *usernum*

Allows you to initiate a disconnect for the specified user. This is similar to option 1 of the PROSHUT menu. Use of this parameter is limited to local non-networked connections only.

-F

Starts an emergency shutdown. To use this parameter, you must run PROSHUT on the machine where the server resides. This parameter is not applicable for remote shutdowns or DataServer shutdowns.

-Gw

For DataServers, specifies the DataServer broker to shut down.

-H *host-name*

Specifies the machine where the database server runs. If issuing the shutdown command from a remote machine, specify the host name.

-S *service-name*

Specifies the database server or broker process. If issuing the shutdown command from a remote machine, specify the service name.

-shutdownTimeout **[** immed **|** maximum **|** *n* **[** h **|** m **|** s **]]**

Specifies the amount of time for normal database activity to stop and users be disconnected before immediately shutting down (killing users still connected). If all normal activity ceases before the timeout value is reached, normal shutdown proceeds. If not specified, the default is 10 minutes. The table below describes the shutdown timeout parameter.

Table 67: shutdownTimeout values

Value	Meaning	Range
immed	Shutdown occurs in approximately 60 seconds, regardless of transaction backout status	–
maximum	Shutdown occurs in a maximum of 24 hours, after all no	–

Value	Meaning	Range
n	Specifies the shutdown timeout value in seconds	60 - 86400
*n*s	Specifies the shutdown timeout value in seconds	60 - 86400
*n*m	Specifies the shutdown timeout value in minutes	1 - 1440
*n*h	Specifies the shutdown timeout value in hours	1 - 24

The -shutdownTimeout parameter is ignored when forced shutdown (-F) is specified.

-cpinternal *codepage*

An internationalization startup parameter that identifies the code page used in memory.

-cpstream *codepage*

An internationalization startup parameter that identifies the code page used for stream I/O.

When you enter the PROSHUT command without the -by, -bn, or -F parameters, the following menu appears:

```
1 Disconnect a User
2 Unconditional Shutdown
3 Emergency Shutdown (Kill All)
x Exit
```

The following table lists the menu options and their actions:

Option	Action
1	Prompts you for the number of the user you want to disconnect.
2	Disconnects all users and shuts down the database.
3	Prompts you to confirm your choice. If you cancel the choice, you cancel the shutdown. If you confirm the choice, PROSHUT displays the following message: Emergency shutdown initiated . . . PROSHUT marks the database for abnormal shutdown, kills all remaining processes connected to the database, and deletes shared-memory segments and semaphores. The database is in a crashed state. PROSHUT performs normal crash recovery when you restart the database and backs out any active transactions.
x	Cancels the shutdown without taking any action.

Notes

- You can shut down using the PROMON utility's **Shut Down Database** menu option.
- The user who shuts down the server must have started it, or be root (on UNIX).

- When you initiate PROSHUT over a network, the amount of time that it takes to actually shut down all of the OpenEdge processes and to free any ports varies depending on the number of clients, brokers, and servers that must be shut down. The PROSHUT command might return control to the terminal before all of the processes are stopped and resources are freed.
- Specifying `-shutdownTimeout` allows the database a specified amount of time to halt activity before initiating immediate database shutdown. The database is considered "active" if any of the following is true:
  - There is forward movement of the before-image "cursor" indicating update activity.
  - There are active connections
  - There are live transactions that have not been backed out

  Normal shutdown proceeds until all activity ceases or the specified amount of time elapses. At the end of the timeout interval, the database is immediately shutdown by stopping all database output and killing all active users.
- If you specified a unique value for `-cpinternal` or `-cpstream` when you opened the database, you must specify that same value for `-cpinternal` or `-cpstream` when you close the database with the PROSHUT command. If you do not, PROSHUT uses the values for `-cpinternal` and `-cpstream` found in the main startup parameter file created during installation (such as *`OpenEdge-install-dir`*`/startup.pf`). If the values of `-cpinternal` or `-cpstream` specified for your database do not match the values specified in the main startup parameter file, you receive the following error message:

  ```
  Code page conversion table for table-name to table-name was not found. (6063)
  ```

- Unconditional forced shutdown (`-byF`) is not allowed on a database started with no integrity (`-i`).
- Forced shutdown (`-F`) on a database started with no integrity (`-i`) requires two confirmations. The first confirmation requires you to acknowledge that the database was started with no integrity and you are initiating a forced shutdown. The second confirmation is the normal forced shutdown confirmation. You must answer `y` to both for the shutdown to execute. This type of shutdown can cause database corruption.

PROWDOG command

Starts the OpenEdge Watchdog process.

Syntax

```
prowdog db-name
```

Parameters

db-name

Specifies the database to clean up after.

The Watchdog cleans up after improperly terminated processes. For example, it releases any locks or shared-memory structures that those processes might hold.

Notes

- If the Watchdog finds a process that is no longer active, it releases all the appropriate record locks, backs out any live transactions, releases any shared-memory locks, and closes the connection. If the lost process is a server, it disconnects and cleans up all appropriate remote clients.
- If the process was changing shared memory when it terminated, shared memory is in an inconsistent state; the Watchdog forces a shutdown to protect the database.
- The Watchdog cannot detect lost remote clients because remote clients are not associated with a process. Instead, a network protocol timeout mechanism notifies the server that the network connection was lost.

23

Database Startup Parameters

This chapter describes OpenEdge database server startup parameters. They are presented in quick reference tables in the beginning of this chapter. Then, each startup parameter is described in detail and listed alphabetically by syntax. The syntax of the parameters is the same for UNIX and Windows unless otherwise noted.

For details, see the following topics:

- Issuing startup parameters
- Database server performance parameters
- Database server-type parameters
- Database server internationalization parameters
- Database server statistics parameters
- Database server consistency check parameters
- Database server SSL connection parameters
- Database server network parameters
- Startup parameter usage categories
- Alphabetical listing of database startup parameters

Issuing startup parameters

You can change the default startup parameters by using startup parameters on a command line, or by incorporating them into a script. You can also use a *parameter file*.

This is a test of the nodmadm conditional.

A parameter file can include any number of startup parameters. This is especially useful if you regularly use the same parameters, or if more parameters are required than can easily fit on the command line. To identify a parameter file, you use the Parameter File (`-pf`) parameter, which has the following syntax:

```
-pf filename
```

You can also include the `-pf` parameter in the parameter file to reference another parameter file.

Note: If duplicate startup parameters are read from the startup line or `.pf` file, the last duplicate parameter read takes precedence.

Database server performance parameters

Use the parameters listed in the table below to optimize server performance.

Table 68: Server performance parameters

Parameter	Syntax	Purpose
After-image Buffers	`-aibufs` *n*	Specifies the number of after-image buffers when running AIW
After-image Stall	`-aistall`	Suspends database activity when an empty after-image (AI) file is unavailable
Blocks in Database Buffers	`-B` *n*	Specifies the number of blocks in the database buffers
Blocks in Database Alternate Buffer Pool	`-B2` *n*	Specifies the number of blocks in the database Alternate Buffer Pool.
Before-image Buffers	`-bibufs` *n*	Specifies the number of before-image buffers when running BIW
Threshold Stall	`-bistall`	Quiets a database and sends a message to the log file when the recovery log threshold is reached; use with `-bithold`
Recovery Log Threshold	`-bithold` *n*	Specifies the maximum size of the recovery log files in MB
Direct I/O	`-directio`	Opens all files in unbuffered mode

Parameter	Syntax	Purpose
Event Level	`-evtlevel`	Specifies the level of information written to the Windows Application Event Log
Before-image Cluster Age	`-G` *n*	Specifies the number of seconds before the database reuses a before-image cluster
Group Delay	`-groupdelay` *n*	Specifies the number of milliseconds a transaction waits before committing
Hash Table Entries	`-hash`	Specifies the number of hash table entries for the buffer pool
No Crash Protection	`-i`	Runs the database without integrity or recovery
Increase parameters online	`-increaseto`	Increases certain predefined server startup parameters while the database is online
Lock Governor	`-LGovernor` *n*	Specifies the percentage of the lock table that a tenant can occupy **Note:** This parameter is applicable to only the default tenant and regular tenant in multi-tenant databases; this parameter does not apply to a super-tenant
Lock Table Entries	`-L` *n*	Specifies the number of entries in the record locking table
Lock release	`-lkrela`	Use the original lock release mechanism installed from previous OpenEdge versions
LRU force skips	`-lruskips` *n*	Specify the number of times a buffer in the buffer pool is accessed before it is placed on the Most Recently Used (MRU) end of the LRU chain
LRU alternate buffer pool force skips	`-lru2skips` *n*	Specify the number of times a buffer in the alternate buffer pool is accessed before it is placed on the Most Recently Used (MRU) end of the LRU chain
Maximum area number	`-maxAreas` *n*	Specify the maximum area number available
Delayed BI File Write	`-Mf` *n*	Delays writing the last before-image (BI) file records
Shared-memory Overflow Size	`-Mxs` *n*	Replaces the default value of the shared-memory overflow area
Number of Users	`-n` *n*	Specifies the maximum number of users connected to the database

Parameter	Syntax	Purpose
Nap maximum	`-napmax` *n*	Specifies the maximum time (in milliseconds) to sleep (nap) after the `-spin` value is exhausted due to failure to acquire a latch
Login Governor	`-nGovernor` *n*	Specifies the maximum number of users that may login for a tenant **Note:** This parameter is applicable to only the default tenant and regular tenant in multi-tenant databases; this parameter does not apply to a super-tenant
Message wait	`-Nmsgwait` *n*	Specifies the number of seconds a server waits for a remote network message
Pin Shared Memory	`-pinshm`	Prevents the OS from swapping shared memory contents to disk
Prefetch delay	`-prefetchDelay`	Enable a delay when sending the first network message for queries with prefetch capabilities
Prefetch factor	`-prefetchFactor` *n*	Specify a percentage of a network message required to contain prefetched data
Prefetch num recs	`-prefetchNumRecs` *n*	Specify the number of prefetch records placed in a network message
Prefetch priority	`-prefetchPriority` *n*	Specify the number of prefetch records to add to an in-process query without polling
Schema lock wait queue	`-schlockwq`	Alters the schema locking algorithm so that the schema lock scheduler forces a shared schema lock request to wait when an exclusive schema lock request is queued
Semaphore Sets[11]	`-semsets`*n*	Changes the number of semaphore sets available to a broker
Shared memory segment size	`-shmsegsize` *n*	Specify the maximum shared memory segment size
Spin Lock Retries	`-spin` *n*	Specifies the number of times a process tries to acquire a latch before pausing
Transaction End (TXE) Retry Limit	`-TXERetryLimit` *n*	Specifies the number of times a connection will re-try to obtain TXE lock after a 1ms nap before getting queued

[11] UNIX only.

Parameter	Syntax	Purpose
Authorized Data Truncation	`SQLTruncateTooLarge` *`value`*	Can be set to *`ON`* or *`ALL`* to truncate data exceeding the column size. If the parameter is set to *`OFF`*, no data is truncated and SQL displays an error. If the parameter is set to *`OUTPUT`*, only the output is truncated. SQL also displays an error if the SQLTruncateTooLarge parameter is not specified at all in case of data exceeding the column size. The default value for the SQLTruncateTooLarge parameter is *`OFF`*.
Autonomous Schema Update	`-SQLWidthUpdate` *`value`*	Helps resolve the SQL width problem by updating the SQL column width in schema automatically when Authorized Data Truncation(ADT) occurs for data in a column. To use this feature, provide the value of the -SQLWidthUpdate parameter as *`ON`* during startup. Once the value of the parameter is set, it is remembered for the lifetime of the server and for all connections. If the value of the -SQLWidthUpdate parameter is set to *`ON`* during startup, the Authorized Data Truncation(ADT) parameter is also enabled irrespective of whether ADT was enabled during startup unless specified otherwise using connection URL. **Note:** This parameter can be provided only during server startup and not in the client connection URL.
Lock Wait Timeout	`-SQLLockWaitTimeout`	Use this parameter (-SQLLockwaitTimeout) to specify the number of seconds to wait when a lock conflict occurs. This value applies to all conflicts that occur while using SQL applications. The default value is five seconds. **Note:** Progress Software Corporation recommends changing the value of Lock Wait Timeout(-SQLLockwaitTimeout) if your installation encounters a lot of conflicts.
Server-side JOIN	`-ssj`	Use this parameter to indicate if the database server will process Server-side JOIN requests from remote clients.

Parameter	Syntax	Purpose
Threaded Server	`-threadedServer`	Use this parameter to indicate a server or servers spawned by the broker uses threads, one per remote client. The default is enabled.
Threaded Server Stack	`-threadedServerStack`	Use this parameter to specify the size of a stack allocated for each thread spawned by a threaded server.

Database server-type parameters

Use the parameters listed in the table below to start a particular type of server.

Table 69: Server-type parameters

Parameter	Syntax	Purpose
Auto Server	`-m1`	Starts an auto server. Used internally by the Database broker
Secondary Login Broker	`-m3`	Starts a secondary broker

Database server internationalization parameters

Use the parameters listed in the table below to control the format in which data appears.

Table 70: Server internationalization parameters

Parameter	Syntax	Purpose
Conversion Map	`-convmap` *filename*	Identifies the conversion map file
Case Table	`-cpcase` *tablename*	Identifies the case table that establishes case rules for the code page
Collation Table	`-cpcoll` *tablename*	Identifies a collation table to use with the code page
Internal Code Page	`-cpinternal` *codepage*	Identifies the code page used in memory
Log File Code Page	`-cplog` *codepage*	Identifies the code page used for writing messages to the log file
Print Code Page	`-cpprint` *codepage*	Identifies the code page used for printer output
R-code in Code Page	`-cprcodein` *codepage*	Identifies the code page used for reading r-code text segments

Parameter	Syntax	Purpose
Stream Code Page	`-cpstream` *codepage*	Identifies the code page used for stream I/O
Terminal Code Page	`-cpterm` *codepage*	Identifies the code page for character terminals

Database server statistics parameters

Use the parameters listed in the table below to collect statistics for table and index access.

Table 71: Server statistics collection parameters

Parameter	Syntax	Purpose
Base Index	`-baseindex` *n*	Specifies a range of indexes for which you want to collect statistics; use with `-indexrangesize`
Base Table	`-basetable` *n*	Specifies a starting table number in a range of tables for which you want to track access statistics; use with `-tablerangesize`
Index Range Size	`-indexrangesize` *n*	Specifies the number of indexes to track for access statistics
Table Range Size	`-tablerangesize` *n*	Specifies the number of tables for which you want to collect statistics

Database server consistency check parameters

Use the parameters listed in the table below to enable consistency checking.

Table 72: Server consistency check parameters

Parameter	Syntax	Purpose
Area consistency check	`-AreaCheck` *area-name*	Area consistency check enables consistency checking for all record and index blocks in the specified area
Database consistency check	`-DbCheck`	Database consistency check enables consistency checking for all record and index blocks in the entire database
Memory consistency check	`-MemCheck`	Memory consistency check enable memory overwrite checking for the buffer pool

Parameter	Syntax	Purpose
Index consistency check	`-IndexCheck` *index-name*	Index consistency check enables consistency checking for all index blocks in the specified index
Table consistency check	`-TableCheck` *table-name*	Table consistency check enables consistency checking for all record blocks of the specified table (excluding BLOBS)

Database server SSL connection parameters

Use the parameters listed in the table below to control the use of SSL.

Table 73: Database server SSL parameters

Parameter	Syntax	Purpose
SSL	`-ssl`	Requires that all brokers and all connections use SSL **Note:** SSL incurs heavy performance penalties, depending on the client, server, and network resources and load.
Key Alias	`-keyalias` *key-alias-name*	Specifies the alias name of the SSL private key/digital certificate key-store entry to use
Key Alias Password	`-keyaliaspasswd` *key-alias-password*	Specifies the encrypted password to the key alias
No Session Cache	`-nosessioncache`	Disables the use of SSL session caching
Session Timeout	`-sessiontimeout` *n*	Specifies the length of time in seconds that an SSL session is held in the session cache

Database server network parameters

Use the parameters listed in the table below to supply the broker with necessary network information.

Table 74: Server network parameters

Parameter	Syntax	Purpose
AdminServer Port	`-adminport` { *service-name* | *port* }	Connects a server group and an AdminServer

Parameter	Syntax	Purpose
SQL Server Java™ Classpath	`-classpath` *pathname*	Identifies the Java™ classpath to use when starting an SQL server
Enforce Mm	`-EnforceMm`	Specifies pre-11.6 behavior where -Mm values must agree for client and server
Host Name	`-H` *host-name*	Specifies a remote host
Maximum Clients Per Server	`-Ma` *n*	Specifies the maximum number of remote users per database server
Maximum Dynamic Server	`-maxport` *n*	Specifies the highest accessible port number in a specified range
Minimum Clients Per Server	`-Mi` *n*	Specifies the number of remote users on a server before a broker starts another server
Minimum Dynamic Server	`-minport` *n*	Specifies the lowest accessible port number in a specified range
Message Buffer Size	`-Mm` *n n*	Specifies the message buffer size for network client/server protocols
Maximum Servers	`-Mn` *n*	Specifies the maximum number of remote client servers that a broker can start
Servers Per Protocol	`-Mp` *n*	Specifies the maximum number of servers to serve remote users for a protocol
Maximum Servers Per Broker	`-Mpb` *n*	Specifies the maximum number of servers that multiple brokers can start to serve remote users for a protocol
Network Type	`-N` *network-type*	Identifies the network communications protocol
Pending Connection Time	`-PendConnTime` *n*	Specifies the amount of time a client has to connect to a server before the broker clears the client's reservation
Configuration Properties File	`-properties` *filename*	Identifies the properties file an AdminServer uses when starting a database server or server group
Service Name	`-S` { *service-name* | *port-number* }	Specifies the service or port number to be used by a broker process

Parameter	Syntax	Purpose
Server Group	`-servergroup` *name*	Identifies a logical collection of server processes to start
Century Year Offset	`-yy` *n*	Determines the start of the 100-year period in which a date with a two-digit year is valid

Startup parameter usage categories

Startup parameters are organized into usage categories. the table below describes each usage category.

Table 75: Startup parameter categories

Usage type	Purpose
Client Session (CS)	Starts a client session
Client Connections (CC)	Connects to a specific database
OpenEdge Database Server (DBS)	Starts a server or broker
DataServer (DS)	Starts non-OpenEdge DataServers

Note: Some parameters fall into more than one category.

Alphabetical listing of database startup parameters

This section describes the OpenEdge database server (DBS) and client connection (CC) startup parameters in detail, in alphabetical order by syntax. Each description begins with a table that shows the syntax of the parameter, then provides other information about it.

For a complete list of startup parameters, see *OpenEdge Deployment: Startup Command and Parameter Reference*.

AdminServer Port (-adminport)

Operating system and syntax	UNIX / Windows	`-adminport` ***service-name port***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

service-name

The name of the service to be used by the AdminServer.

port

The port number the AdminServer uses to communicate with server groups. The default port is 7844.

Use AdminServer Port (`-adminport`) to establish communication between a server group and an AdminServer. The AdminServer uses this parameter internally. The `-adminport` setting must match the `-admin` setting specified when the AdminServer was started.

After-image File Management Archive Directory List (-aiarcdir)

Operating system and syntax	UNIX / Windows	**-aiarcdir *dirlist***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

dirlist

A comma separated list of directories where archived after-image files are written by the AI File Management Utility. The directory names can not have any embedded spaces.

Use After-image File Management Archive Directory List (`-aiarcdir`) to specify the directories where archived after-image files are written by AI File Management. The directories must exist, unless you also specify `-aiarcdircreate` to direct the utility to create the directories. This parameter is useful only when running the AI File Management utility.

After-image File Management Archive Directory Create (-aiarcdircreate)

Operating system and syntax	UNIX / Windows	**-aiarcdircreate**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

Use After-image File Management Archive Directory Create (`-aiarcdircreate`) to direct the AI File Management utility to create the directories specified by `-aiarcdir`. This parameter is useful only when running the AI File Management utility.

After-image File Management Archive Interval (-aiarcinterval)

Operating system and syntax	UNIX / Windows	-aiarcinterval *n*		
Use with	**Maximum value**	**Minimum value**[12]	**Single-user default**	**Multi-user default**
DBS	86400	120	–	120

n

The number of seconds between mandatory extent switches by AI File Management. Omit `-aiarcinterval` for on-demand mode.

Use After-image File Management Archive Interval (`-aiarcinterval`) to specify timed or on-demand mode for the operation of AI File Management. If you specify timed mode, *n* sets the elapsed time in seconds between forced AI extent switches. The minimum time is two minutes, the maximum is 24 hours. If you omit this parameter to indicate on-demand mode, there are no mandatory extent switches other than when an extent is full. You can modify the interval on a running system with RFUTIL. Regardless of the mode, AI File Management archives full AI extents every five seconds. This parameter is useful only when running the AI File Management utility.

After-image Buffers (-aibufs)

Operating system and syntax	UNIX / Windows	-aibufs *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	1	–	20

n

The number of after-image buffers.

Use After-image buffer (`-aibufs`) to specify the number of after-image buffers. This parameter is useful only when running the after-image writer (AIW) because the AIW writes the filled after-image buffers to disk, making the buffers available to other client and server processes. Try setting `-aibufs` to a value of 1.5 times the value of the Before-image Buffers (`-bibufs`) parameter, or a minimum of 5.

Without the AIW writing the buffers, any gain from increasing the number of buffers is negligible.

Note: Validations using **-aibufs** can be run online as part of routine health checks. For more information, see PROUTIL Utility on page 629.

---

[12] For timed mode, the minimum value is 120. For on-demand mode, do not specify this parameter.

After-image Stall (-aistall)

Operating system and syntax	UNIX / Windows	-aistall		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

Use After-image Stall (`-aistall`) to suspend database activity if all AI files are filled. `-aistall` ceases all database activity until the AI extent is emptied and sends this message to the log file:

```
Can't switch to after-image extent filename it is full. (3775)
Backup ai extent and mark it as empty (3776)
```

When using after-image (AI) files, monitor the status of the files to ensure that the AI extents do not run out of space and cause the database to hang. Without the use of `-aistall`, the database shuts down when the AI files are filled.

Area consistency check (-AreaCheck)

Operating system and syntax	UNIX / Windows	-AreaCheck *areaname*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	—	—	–	—

areaname

Specifies the name of the area for consistency checking.

Use Area consistency check to enable consistency checking for all record and index blocks in the specified area. When enabled, this option applies consistency checks to all index blocks and record blocks for record and index write operations. You can only specify one area with `-AreaCheck`. To check multiple areas, you must apply consistency checking to the entire database with Database consistency check (-DbCheck) on page 498.

Area consistency check validates a block is still physically correct after an operation has been performed. For example, after an index key is inserted into an index block, the consistency check validates that the block still laid out correctly.

Area consistency checking can be enabled or disabled while your database is online with PROMON. See the Enable/disable block level consistency check on page 1051 for more information.

Area consistency check can be enabled for a single user client or for any offline utility. If the utility is read-only, such as DBANALYS, the `-AreaCheck` parameter is ignored. Online utilities determine whether or not to perform area consistency checking during execution based on the use of `-AreaCheck` at broker startup or by the enablement/disablement of the database consistency checking in PROMON.

Blocks in Database Buffers (-B)

Operating system and syntax	UNIX / Windows	-B *n*		
Use with	**Maximum value**[13]	**Minimum value**	**Single-user default**	**Multi-user default**[14]
CC, DBS	1,000,000,000 or 125,000,000	10	20	Maximum of (8 * *users)* or 3000

n

The number of blocks in the database buffers.

Use Blocks in Database Buffers (-B) to specify the number of blocks in the database buffers. The optimum value depends on your application. Validations to ensure database consistency in both index and data blocks during forward processing.using -B can be run online as part of routine health checks. For more information, see PROUTIL Utility on page 629

Note: On the AIX platform, when starting a database with large shared memory requirements (for instance, when the -B exceeds the allotted system paging space), the system may become unstable if the PSALLOC=early environment variable is not set. For more information, see *OpenEdge Getting Started: Installation and Configuration*.

Blocks in Alternate Buffer Pool (-B2)

Operating system and syntax	UNIX / Windows	-B2 *n*		
Use with	**Maximum value**[15]	**Minimum value**	**Single-user default**	**Multi-user default**[16]
CC, DBS	1,000,000,000 or 125,000,000	10	0	0

n

The number of blocks allocated in the alternate buffer pool.

Use Blocks in Alternate Buffer Pool (-B2) to specify the number of blocks in the alternate buffer pool. The optimum value depends on your application.

You cannot specify Blocks in Alternate Buffer Pool for a Replication target database.

[14] **The *users* value is specified by the Number of Users (-n) parameter.**

[13] **The maximum value of the sum of -B + -B2: 1,000,000,000 for 64-bit platforms; 125,000,000 for 32-bit platforms. The maximum is system dependent and limited by available memory.**

[16] **The *users* value is specified by the Number of Users (-n) parameter.**

[15] **The maximum value of the sum of -B + -B2 . 1,000,000,000 for 64-bit platforms; 125,000,000 for 32-bit platforms. The maximum is system dependent and limited by available memory.**

Base Index (-baseindex)

Operating system and syntax	UNIX / Windows	**-baseindex *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32765	-32765	–	–

n

The starting index number in the range of indexes for which you want to track access statistics.

Use Base Index (`-baseindex`) with Index Range Size (`-indexrangesize`) to specify the range of indexes for which you want to collect statistics. Access to the statistics is handled through the Virtual System Tables (VSTs). Index statistics are stored in the _IndexStat VST. To obtain index numbers, use the following ABL code:

```
FOR EACH _file NO-LOCK:
  DISPLAY _file._file-name.
  FOR EACH _index WHERE _index._file-recid = RECID(_file) NO-LOCK:
    DISPLAY _index._idx-num _index._index-name.
  END.
END.
```

This results in the following output:

```
File-Name
filename

_idx-num Index-Name
      n1   index name1
      n2   index name2
      n3   index name3
```

Base Table (-basetable)

Operating system and syntax	UNIX / Windows	***-basetable n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32765	-32765	–	–

n

The starting table number in the range of tables for which you want to track access statistics.

Use Base Table (-basetable) with Table Range Size (-tablerangesize) to specify the range of tables for which you want to collect statistics. Access to the statistics is handled through the Virtual System Tables (VSTs). Table statistics are stored in the _TableStat VST. To obtain table numbers, use the following ABL code:

```
FOR EACH _file NO-LOCK:
  DISPLAY _file._file-num _file._file.
END.
```

This results in the following output:

```
_File-Number File-Name
      n1    table name1
      n2    table name2
      n3    table name3
```

Base user index (-baseuserindex)

Operating system and syntax	UNIX / Windows	**-baseuserindex *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32765	-32765	1	1

n

The starting index number for the range of indexes for which you want gather information in the _UserIndexStat VST.

Use Base user index (-baseuserindex) in conjunction with User index range (-userindexrangesize) to specify the first index, and the the number of indexes for which statistics are collected in the_UserIndexStat VST.

Note: If the User index base and range are not specified, the _UserIndexStat VST is dimensioned using the values specified for the _IndexStat VST (-baseindex and -indexrangesize).

Base user table (-baseusertable)

Operating system and syntax	UNIX / Windows	**-baseusertable *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32765	-32765	1	1

n

The starting table number for the range of user tables for which you want gather information in the `_UserTableStat` VST.

Use Base user table (`-baseusertable`) in conjunction with User table range (`-usertablerangesize`) to specify the first table, and the the number of tables for which statistics are collected in the `_UserTableStat` VST.

Note: If the User table base and range are not specified, the `_UserTableStat` VST is dimensioned using the values specified for the `_TableStat` VST (`-basetable` and `-tablerangesize`).

Before-image Buffers (-bibufs)

Operating system and syntax	UNIX / Windows	***-bibufs n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	3	–	20

n

The number of before-image buffers.

Use Before-image Buffers (`-bibufs`) to specify the number of before-image buffers. This parameter is useful only when running the before-image writer (BIW). The BIW continually writes the filled before-image buffers to disk, making the buffers available to other client and server processes. Without a BIW writing the buffers, any gain from increasing the number of buffers is negligible.

Note: Validations using **-bibufs** can be run online as part of routine health checks. For more information, see PROUTIL Utility on page 629.

Threshold Stall (-bistall)

Operating system and syntax	UNIX / Windows	**-bistall**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

Use Threshold Stall (`-bistall`) with Recovery Log Threshold (`-bithold`) to quiet the database when the recovery log threshold is reached, without performing an emergency shutdown. When you use `-bistall`, a message is added to the database log (`.lg`) file stating that the threshold stall is enabled.

Recovery Log Threshold (-bithold)

Operating system and syntax	UNIX / Windows	-bithold *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	System dependent[17]	System dependent[17]	–	–

n

An integer specifying the threshold, in MB.

Use Recovery Log Threshold (`-bithold`) to set the maximum size to which recovery log files can grow. The recommended threshold is between three and one hundred percent (3-100%) of the largest possible recovery log file size, rounded to the nearest cluster boundary. Once the threshold is reached, the database performs an emergency shutdown. Using the `-bistall` parameter with `-bithold` prevents the emergency shutdown; instead, the database ceases activity until the BI extent is emptied.

Schema Cache File (-cache)

Operating system and syntax	UNIX Windows	-cache *filename*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC	–	–	–	–

filename

The pathname of a binary schema cache file.

Use Schema Cache File (`-cache`) to read the database schema from a local file instead of the database. You must have previously built the schema cache and stored it as a binary file.

To perform database activities, the OpenEdge client keeps a copy of the database schema called the *schema cache* in memory. By default, OpenEdge creates the schema cache by reading the database schema stored in the database file. The time required to read the schema usually is minimal; however, under the following conditions, the time required to read the schema might be unacceptable:

- If the client connects to the database over a wide-area network (WAN)
- When a large number of clients connect to a database simultaneously, for example, after a database shutdown or crash

Connection time depends on several factors, including schema size.

To reduce connection time, OpenEdge lets you store the schema cache as a binary file, called a *schema cache file*, on a local disk. The client can then read the schema directly from the schema cache file.

[17] Limited by available disk space.

To write the schema cache file, you build the desired schema cache and save it to a binary file using the ABL SAVE CACHE statement. The schema cache file is portable across systems, so you can create the file once and distribute it across a heterogeneous network of systems. For information on building and saving the schema cache file, see *OpenEdge Development: Programming Interfaces*.

If you specify schema cache file (-cache) when you connect to a database and the local schema is valid, the AVM reads the schema from the local file instead of from the database. The schema cache is valid if the time stamp of the schema cache file matches the time stamp in the database master block. If the time stamps do not match, or for some reason the AVM cannot read the file, the AVM issues a warning message and reads the schema from the database.

Note: If you are generating the local binary schema cache, do not connect to the database using Trigger Location (-trig) and Schema Cache File (-cache) together.

CDC cache size (-cdcsize)

Use CDC cache size (-cdcsize) to specify the size of the CDC cache.

Operating system and syntax	**UNIX / Windows**	**-cdcsize *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	1400320	200	2000 if CDC **is** enabled. 200 if CDC **is not** enabled.	2000 if CDC **is** enabled. 200 if CDC **is not** enabled.

n

The size, in kbytes of both the primary and secondary CDC caches combined.

If you specify a value smaller than 200, the size is increased to 200.

In single-user mode, the primary cache is not used, and the entire size is allocated to the secondary cache.

In multi-user mode, 75 percent of the cache size is allocated to the primary cache, and the remainder to the secondary. However for values smaller than 400, the size is split evenly between the two caches.

You can increase the size of the secondary cache at runtime with PROUTIL INCREASETO.

CDC cache size (-cdcsize) is only for databases enabled for Change Data Capture. For best performance, you should set CDC cache size (-cdcsize) large enough that all your existing records fit in the primary cache.

SQL Server Java Classpath (-classpath)

Operating system and syntax	UNIX / Windows	-classpath *pathname*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

pathname

Specifies the pathname of the classpath.

Use SQL Classpath (`-classpath`) to identify the Java classpath to use when starting an SQL server. SQL database brokers use this parameter when launching the Java Virtual Machine (JVM) to execute stored procedures. The default is to use the current environment variable CLASSPATH setting. You do not use this parameter directly.

Cluster Mode (-cluster)

Operating system and syntax	UNIX / Windows	-cluster *qualifier*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

qualifier

Direct a broker to start a cluster-enabled database directly, or to instruct the cluster manager to perform the startup. The table below describes the accepted values for *qualifier*.

Table 76: Values for -cluster

Value	Description
`startup`	Instructs the broker to redirect the startup request to the OS cluster manager. The cluster manager will start the database using the information supplied when the database was enabled for cluster protection.
`protected`	Instructs the broker to start a cluster-enabled database. This value is used by the cluster manager within the database `.pf` file and indicates that database will be protected by the cluster manager in the event of a failover. **Note:** This parameter must only be used by the cluster manager. It should not be used on the command line or a directly executed script.

The -cluster parameter is required for startup of all cluster-enabled databases. If used with a database that is not cluster-enabled, the parameter is ignored.

Conversion Map (-convmap)

Operating system and syntax	UNIX / Windows	-convmap *filename*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS	–	–	–	–

filename

The pathname of your CONVMAP file.

Use Conversion Map (-convmap) to identify the CONVMAP file to use for code page conversions, collation orders, and case conversions. By default, the convmap.cp file in the OpenEdge install directory is used. You can create a CONVMAP file by using the PROUTIL utility with the CODEPAGE-COMPILER qualifier. See *OpenEdge Development: Internationalizing Applications* for more information on CONVMAP files.

Case Table (-cpcase)

Operating system and syntax	UNIX / Windows	-cpcase *tablename*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS	–	–	Basic	Basic

tablename

The name of a case table in the convmap.cp file.

Use Case Table (-cpcase) to specify the case table. This table establishes case rules for the code page used in memory. The code page is specified by the Internal Code Page (-cpinternal) parameter. The case rules are used by the CAPS and LC functions. Also, in a character field format you can use an exclamation point (!) to convert all characters to uppercase during input.

To retrieve the value of this startup parameter at run time, use the SESSION System handle.

Collation Table (-cpcoll)

Operating system and syntax	UNIX / Windows	-cpcoll *tablename*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS	–	–	Basic	Basic

tablename

The name of a collation table within the convmap.cp file.

Use Collation Table (-cpcoll) to identify a collation table to be used with the code page in memory. The code page is specified by the Internal Code Page (-cpinternal) parameter.

The collation rules that you specify are used to compare characters and sort records if a BY clause cannot be satisfied by an index. The collation rules specified with the -cpcoll parameter take precedence over the collation specified for any database accessed during the session, except when pre-existing indexes are used or modified. When you update or rebuild a database's indexes, the collation rules originally defined for that database are used.

If you do not use -cpcoll, the language collation rules defined for the first database on the command line are used. If you do not specify a database on the command line, collation rules with the name "basic" are used.

To retrieve the value of this startup parameter at run time, use the SESSION System handle.

See *OpenEdge Development: Internationalizing Applications* for more information on collation tables.

Internal Code Page (-cpinternal)

Operating system and syntax	UNIX / Windows	**-cpinternal *codepage***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS, DS	–	–	iso8859-1	iso8859-1

codepage

The name of the code page used in memory.

Use Internal Code Page (-cpinternal) to identify the code page used in memory and for graphical clients. For graphical clients, the -cpinternal code page should be the same code page that the operating system uses. If you do not use -cpinternal, the iso8859-1 code page is used by default. To retrieve the value of this startup parameter at run time, use the CPINTERNAL attribute of the SESSION System handle.

Note: Do not use a 7-bit table with -cpinternal. Use 7-bit tables for converting data from a 7-bit terminal to another code page only. Do not use them for character conversion in memory or for the database.

Log File Code Page (-cplog)

Operating system and syntax	UNIX / Windows	**-cplog *codepage***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS, DS	–	–	-cpstream	-cpstream

codepage

The name of the code page for messages written to the log file.

Use Log File Code Page (`-cplog`) to identify the code page used to write messages to the log (`.lg`) file. If you do not specify a value, the default is the code page specified by Stream Code Page (`-cpstream`).

To retrieve value of this startup parameter at run time, use the `CPLOG` attribute of the `SESSION` System handle.

Print Code Page (-cpprint)

Operating system and syntax	UNIX / Windows	**-cpprint *codepage***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS, DS	–	–	-cpstream	-cpstream

codepage

The name of the code page used for printer output.

Use Print Code Page (`-cpprint`) to identify the code page used for printing. When you print a file, the code page specified by `-cpprint` overrides the code page specified by Stream Code Page (`-cpstream`).

To retrieve the value of this startup parameter at run time, use the `SESSION` System handle.

R-code in Code Page (-cprcodein)

Operating system and syntax	UNIX / Windows	**-cprcodein *codepage***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS, DS	–	–	-cpinternal	-cpinternal

codepage

The name of the code page for reading r-code text segments.

Use R-code In Code Page (`-cprcodein`) to read the r-code text segments, as if they were written in the code page specified by `-cprcodein`, and convert them to the Internal Code Page (`-cpinternal`) code page.

Caution: This parameter is for use during very rare situations and, in general, should not be used.

The OpenEdge RDBMS reads text segments as if they are written in the code page specified by -cprcodein, even if the text segments were written with a different code page. For example, if you use the following startup parameters and run a .r file written with code page IBM850, the text segments are converted from ISO8859-1 to ibm861. This can produce incorrect results if the .r file was correctly labeled internally as IBM850:

```
-cprcodein ISO8859-1 -cpinternal ibm861
```

To retrieve the value of this startup parameter at run time, use the SESSION System handle. To determine the code page of an r-code file, use the RCODE-INFO handle.

Stream Code Page (-cpstream)

Operating system and syntax	UNIX / Windows	-cpstream *codepage*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS, DS	–	–	ibm850	ibm850

codepage

The name of the code page for stream I/O.

Use Stream Code Page (-cpstream) to identify the code page used for stream I/O. Character terminals use the code page you specify for -cpstream unless you also specify a value for Terminal Code Page (-cpterm), Print Code Page (-cpprint), or Log File Code Page (-cplog).

Stream I/O consists of the following elements:

- Terminals (includes character terminals and DOS Protected mode, but does not include graphical interfaces or the Windows character interface)
- Data (.d) files
- READ-FILE, WRITE-FILE, and INSERT-FILE methods for the EDITOR widget
- INPUT FROM and OUTPUT TO statements
- All compilable files (for example, .p, .w, .i)
- Compiler-generated LISTING, XREF, and PREPROCESS files

Note: Do not use a 7-bit table with -cpstream. Use 7-bit tables for converting data from a 7-bit terminal to another code page only. Do not use them for character conversion in memory or for the database.

To retrieve the value of this startup parameter at run time, use the SESSION System handle. To determine the code page of an r-code file, use the RCODE-INFO handle.

Terminal Code Page (-cpterm)

Operating system and syntax	UNIX / Windows	-cpterm *codepage*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS, DS	–	–	-cpstream	-cpstream

codepage

The name of the code page for character terminals.

Use Terminal Code Page (`-cpterm`) to identify the code page of your character terminals. This parameter allows you to specify a different code page for character terminals than used by the rest of stream I/O, which is set by Stream Code Page (`-cpstream`).

Note: You can use a 7-bit table with `-cpterm`.

To retrieve the value of this startup parameter at run time, use the `SESSION` System handle. To determine the code page of an r-code file, use the `RCODE-INFO` handle.

Crash Recovery Status (-crStatus)

Operating system and syntax	UNIX / Windows	-crStatus *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, CS, DBS	1800	0	30	30

n

The time, in seconds, between crash recovery progress messages.

Specify 0 to turn off crash recovery progress messages.

Use Crash Recovery Status(`-crStatus`) to specify how often a crash recovery progress message is generated during crash recovery.

Each crash recovery progress message contains the following details:

- The current crash recovery phase
- An estimate of the progress of the current crash recovery phase, expressed as the percent complete
- The current block being processed
- The total number of elapsed seconds for the current phase
- Optionally, if `-crTXDisplay` is also specified, the counter of the BI cluster. For more information, see Crash Recovery Transaction Display (-crTXDisplay) on page 498.

For more information about crash recovery, see Crash recovery on page 118.

Crash Recovery Transaction Display (-crTXDisplay)

Operating system and syntax	UNIX / Windows	-crTXDisplay		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, CS, DBS	–	–	–	–

Use Crash Recovery Transaction Display (`-crTXDisplay`) to display to the screen only (not the log file) the contents of the transaction table prior to the Physical Undo phase of crash recovery. When `-crTXDisplay` is specified, the progress messages of crash recovery, controlled by `-crStatus`, include the counter of the current BI cluster. Use this information to cross-reference the transaction table to determine the location in the BI file of the current phase, relative to the remaining transactions. This information can assist you in determining if there are many transactions to back out during crash recovery, or if there is perhaps one long-running transaction.

The transaction display message contains the following:

- The transaction ID
- The timestamp when the transaction began
- The user number that initiated the transaction
- The counter of the BI cluster that was current when the transaction began

For more information about crash recovery progress messages, see Crash Recovery Status (-crStatus) on page 497.

For more information about crash recovery, see Crash recovery on page 118.

Database consistency check (-DbCheck)

Operating system and syntax	UNIX / Windows	-DbCheck		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	–	–	–	–

Use Database consistency check to enable consistency checking for all record and index blocks in the entire database. When enabled, this option applies consistency checks to all index blocks and record blocks for record and index write operations.

Database consistency check validates a block is still physically correct after an operation has been performed. For example, after an index key is inserted into an index block, the consistency check validates that the block still laid out correctly.

You can limit the scope of consistency checking by specifying one area with Area consistency check (-AreaCheck) on page 485, one table with Table consistency check (-TableCheck) on page 536, or one index with Index consistency check (-IndexCheck) on page 503.

Database consistency checking can be enabled or disabled while your database is online with PROMON. See Enable/disable block level consistency check on page 618 for more information.

Database consistency check can be enabled for a single user client or for any offline utility. If the utility is read-only, such as DBANALYS, the `-DbCheck` parameter is ignored. Online utilities determine whether or not to perform database consistency checking during execution based on the use of `-DbCheck` at broker startup or by the enablement/disablement of the database consistency checking in PROMON.

Direct I/O (-directio)

Operating system and syntax	UNIX / Windows	**-directio**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	–	–	Not enabled	Not enabled

Use Direct I/O (`-directio`) to open all files in unbuffered mode, which enables the OpenEdge RDBMS to use an I/O technique that bypasses the operating system buffer pool and transfers data directly from a buffer to disk. This technique avoids the overhead of maintaining the operating system buffer pool and eliminates competition for operating system buffers between OpenEdge programs and other programs.

You might improve performance by using the direct I/O feature. To use direct I/O, use Blocks in Database Buffers (`-B`) to increase the size of the buffer pool, since OpenEdge database I/O will not pass through the operating system buffer pool. Also, decrease the size of the operating system buffer pool to compensate for the additional memory allocated to the database.

Note: Use asynchronous page writers (APWs). They improve database performance by performing overhead operations in the background.

Encryption cache size (-ecsize)

Use Encryption Cache Size (`-ecsize`) to specify the size (in number of encryption policies) of the caches for encryption policies.

Operating system and syntax	UNIX / Windows	**-ecsize *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS, CC	-	0	1000	1000

n

The size, in number of cache entries, of the primary and secondary encryption caches.

For Transparent Data Encryption and Enterprise RDBMS licenses only.

The encryption cache holds your encryption policies in memory, eliminating the need to read policy records from the database during the encryption and decryption processing.

At startup, if your database is currently enabled for transparent data encryption, two encryption caches are allocated, a primary cache and a secondary cache. If your database is not enabled for transparent data encryption, only the secondary cache is allocated. Each cache is initially the size specified by Encryption Cache Size (`-ecsize`).

Existing encryption policies are cached in the primary encryption cache. If the primary cache is filled, remaining encryption policies are inserted into the secondary cache. All new encryption policies are cached in the secondary cache. If the secondary cache is filled, an informational message is written to the database log file, and the cache is expanded by 1000. If at any time the secondary cache cannot be expanded, the database shuts down.

You should set Encryption cache size (`-ecsize`) large enough that all your existing encryption policies fit in the primary encryption cache.

You can set Encryption cache size (`-ecsize`) to 0 (zero) if you never intend to enable encryption on your database.

See *OpenEdge Data Management: Database Administration* for more information on transparent data encryption.

Enforce Mm (-enforceMm)

Operating system and syntax	UNIX / Windows	**-enforceMm**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	—	—	—	—

Use Enforce Mm (`-enforceMm`) to specify that the behavior of the Message Buffer Size (`-Mm`) startup parameter follow prior releases.

OpenEdge uses message buffers to move records (messages) between servers and remote clients. The size of the message buffer is specified with `-Mm`. Starting in OpenEdge Release 11.6, the value of `-Mm` is not required to agree between the server and the client, however in releases prior to OpenEdge 11.6, the value of `-Mm` had to be specified the same for both the client and the server or the connection is refused. Use Enforce Mm (`-enforceMm`) to require agreement between the client and server values of `-Mm` as in earlier releases.

Event Level (-evtlevel)

Operating system and syntax	Windows	**-evtlevel *value***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS, CS	–	–	Normal	Normal

value

Use Event Level (-evtlevel) to specify the level of information written to the Windows Application Event Log. Valid values include:

- **None** — No events are written to the Event Log.
- **Brief** — Error and Warning messages are written to the Event Log.
- **Normal** — Error and Warning messages are written to the Event Log along with any message that is normally written to the log file (`.lg`). This is the default value.
- **Full** — Error, Warning, and Informational messages are written to the Event Log along with any messages generated by the Message Statement.

For more information about using the Event Log, see Logged Data on page 439.

Before-image Truncate Interval (-G)

Operating system and syntax	UNIX / Windows	-G *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32	0	0	0

n

The number of seconds the database engine waits.

Use Before-image Truncate Interval (`-G`) to specify the number of seconds before the database engine reuses a before-image cluster.

Note: In earlier releases, this parameter was named Before Image Cluster Age.

Group Delay (-groupdelay)

Operating system and syntax	UNIX / Windows	-groupdelay *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	1,000	0	0	1

n

The number of milliseconds a transaction waits before committing.

Use Group Delay (`-groupdelay`) to increase performance when Delayed BI File Write (`-Mf`) is set to zero. When the Group Delay is set greater than zero (0), the database engine uses a technique known as Group Commit. When using Group Commit, a transaction spools its end note to the BI buffer and waits a short time until the buffer becomes full and is written to disk, or waits for other transactions to end and store their end notes in the BI buffer so that several transactions are committed by the same synchronous write. In this manner, Group Commit benefits overall performance, although each individual transaction might take slightly longer.

Host Name (-H)

Operating system and syntax	UNIX / Windows	-H { *host-name* | localhost[18] }		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS, DS	–	–	–	–

host-name

The name (address) of the database server machine. This name is assigned to the machine in your TCP/IP hosts file.

localhost

A reserved word that specifies that the Database server communicates only with clients on the database server machine.

Use Host Name (-H) to identify the host name.

Hash Table Entries (-hash)

Operating system and syntax	UNIX / Windows	-hash *n*	
Use with	**Maximum value**	**Minimum value**	**Default**
CC, DBS	–	13	Approximately 1/4 of the -B value

n

The number of hash table entries to use for the buffer pool.

Do not use the **-hash** parameter unless directed to do so by Technical Support.

No Crash Protection (-i)

Operating system and syntax	UNIX / Windows	-i	
Use with	**Maximum value**	**Minimum value**	**Default**
CC, DBS	–	–	–

[18] **localhost does not apply to DataServers.**

Use No Crash Protection (`-i`) to run the database without integrity or database recovery. When running without database integrity, the database engine writes fewer data and before-image blocks to the disk. In this mode, some procedures (such as those that create and delete large numbers of records) can run significantly faster than if they are running with database integrity.

When running with the `-i` parameter, transaction undo is supported. Therefore, there will still be a before-image file, which might grow quite large during very long transactions.

Use this parameter to do bulk data loading or for large batch runs. It reduces the number of disk input or output operations. Loading a database for the first time is a good example of a use for this parameter.

Caution: If you run your database with the `-i` parameter and the database fails for any reason, you cannot recover the database.

Do not use the `-i` parameter unless you have a complete backup of the database and can rerun procedures in case of a system failure. If the system fails during an OpenEdge session that started without crash protection, restore the backup copy and rerun the necessary procedures. For information on restoring a database, see Recovering a Database on page 117.

Index consistency check (-IndexCheck)

Operating system and syntax	UNIX / Windows	`-IndexCheck { tablename.indexname | ownername.tablename.indexname | ownername.tablename.indexname.partitionname }`		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	–	–	–	–

ownername

Specifies the owner of the table containing the index you want to check. You must specify an owner name unless the table's name is unique within the database, the table is owned by PUB, or if you are specifying a partition of the index. By default, ABL tables are owned by PUB.

tablename

Specifies the table containing the index to be checked.

indexname

Specifies the name of the index for consistency checking.

partitionname

Specifies the partition of the index for consistency checking.

Use Index consistency check to enable consistency checking for all index blocks in the specified index. When enabled, this option applies consistency checks to all index blocks and index write operations.

Index consistency check validates a block is still physically correct after an operation has been performed. For example, after an index key is inserted into an index block, the consistency check validates that the block still laid out correctly.

You can only specify one index or partition with `-IndexCheck`. To check multiple partition, apply consistency checking to the entire index. To check multiple indexes, you can apply consistency checking to the entire database with Database consistency check (-DbCheck) on page 498, or if multiple indexes are in one area, you can apply consistency checking to an entire area with Area consistency check (-AreaCheck) on page 485.

When specifying the index or index partition to check, the following parsing rules are applied:

- If two variables are supplied, they are interpreted as the table name and the index name (*tablename.indexname*).
- If three variables are supplied, they are interpreted as the ownername, the table name, and the index name (*ownername.tablename.indexname*).
- Four variables **must** be supplied to specify a partition of an index (*ownername.tablename.indexname.partitionname*).

Index consistency checking can be enabled or disabled while your database is online with PROMON. See Enable/disable block level consistency check on page 618 for more information.

Index consistency check can be enabled for a single user client or for any offline utility. If the utility is read-only, such as DBANALYS, the `-IndexCheck` parameter is ignored. Online utilities determine whether or not to perform index consistency checking during execution based on the use of `-IndexCheck` at broker startup or by the enablement/disablement of the index consistency checking in PROMON.

Index Range Size (-indexrangesize)

Operating system and syntax	UNIX / Windows	**-indexrangesize *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

n

The number of indexes for which you want to track access statistics.

Use Index Range Size (`-indexrangesize`) to specify the number of indexes for which you want to collect statistics from virtual system tables (VSTs). See Virtual System Tables on page 863 for more information on VSTs.

Internet Protocol (-ipver)

Use Internet Protocol (`-ipver`) to identify the IP version for network connections.

Operating system and syntax	UNIX Windows	**-ipver { IPv4 | IPv6 }**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Client Connection, Database Server	–	–	–	`IPv4`

`IPv4`

Specifies Internet Protocol Version 4. Only network connections with IPv4 are allowed.

`IPv6`

Specifies Internet Protocol Version 6. IPv6 allows network connections with IPv6 addresses and mapped IPv4 addresses. OpenEdge supports V4 mapped addresses where underlying operating system support exists: UNIX, Linux, and Windows Vista.

Note: The `-ipver` startup parameter is case sensitive, and must be specified in all lower case. The values `IPv4` and `IPv6` are not case sensitive, and can be specified in any case.

Port mapping for IPv4 network connections behaves differently than IPv6 network connections. When providing a non-local network address using `-H`, IPv4 will bind all the addresses on the system. IPv6 binds only those network addresses specified with `-H`.

These port mapping differences have the following effects:

- Multi-homed systems (those connected to several LANs) need multiple brokers if they need to be accessed by clients from multiple networks. In particular, if you associate `-H` with a non-local address, the client can not connect with `-H localhost`.

  In previous multi-homed systems (those using IPv4) servers were bound to all interfaces on the system (binding occurred for INADDR_ANY). Only one broker supported connections from all LANs to which the system was connected. With IPv6, binding is specific to the addresses specified with `-H`.

- IPv4 and IPv6 are not compatible for environments using a dual protocol database with two brokers listening on both TCP4 and TCP6.

  Starting a server with IPv4 and a non-local network address binds it to all addresses on the system, preventing further binds on the same port with a specific IPv6 address. Conversely, if a server is already bound to a TCP6 address/port, starting up another server on the same port with IPv4 may fail because on some systems (for example, LINUX) a bind to INADDR_ANY reserves the port number in the IPv6 space.

Key Alias (-keyalias)

Operating system and syntax	UNIX Windows	**-keyalias *key-alias-name***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	default_server

key-alias-name

Specifies the alias name of the SSL private key/digital certificate key-store entry to use.

Use Key Alias (`-keyalias`) to identify a SSL private key/digital certificate key-store other than the default.

Key Alias Password (-keyaliaspasswd)

Operating system and syntax	UNIX Windows	**-keyaliaspasswd *key-alias-password***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	password[19]

key-alias-password

Specifies the encrypted SSL Key alias password to use to access the server's private key/digital certificate key-store entry. The default is the encrypted value of `password`.

Use Key Alias Password (`-keyaliaspasswd`) to allow access to the Key Alias when you use a Key Alias other than the default. The *`key-alias-password`* value must be encrypted. You can use the `genpassword` utility, located in your installation's `bin` directory, to encrypt the password.

Lock Governor (-LGovernor)

Operating system and syntax	UNIX Windows	**-LGovernor *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	100	1	–	–

n

The percent of the lock table one tenant can occupy.

[19] The actual value used is the encrypted value of the string "password".

Lock Governor (`-LGovernor`) specifies the percentage of the lock table that a tenant can occupy. Lock governor prevents a tenant from acquiring all the locks in a table. Lock governor applies to the default tenant and regular tenant. This governor does not apply to super-tenants and non multi-tenant databases.

When a tenant attempts to acquire more locks than the governor allows, an error message is displayed and recorded in the database log as:

```
Attempt to exceed tenant lock governor for tenant <tenantID>. Increase -L or -LGovernor.
```

You can set or modify the `-LGovernor` value using the PROMON utility. For more information about the PROMON utility, see PROMON Utility on page 543. To set the `-L` value, see Lock Table Entries (-L) on page 507.

Lock Table Entries (-L)

Operating system and syntax	UNIX / Windows	**-L *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	System dependent[20]	32	–	8192

n

The number of entries in the record locking table. If you specify a value that is not a multiple of 32, the value you specify is rounded to the next highest multiple of 32.

Use Lock Table Entries (`-L`) to change the limits of the record locking table. Each record that is accessed and locked by a user takes one entry. This is true whether the record is accessed with SHARE-LOCK or EXCLUSIVE-LOCK. Increase the size of the lock table if the following message appears:

```
SYSTEM ERROR: Record lock table too small. Increase -L parameter.
```

This message might also indicate that a particular procedure should be restructured into smaller transactions or should be run in single-user rather than multi-user mode. When lock table limits are exceeded, check to make sure transactions are not too large before increasing the lock table size.

If a user process tries to acquire a lock and the lock table overflows, the user's program is aborted, but the server continues to operate. Any partial transactions are undone.

Note that two record locks are acquired when records are accessed with the `BREAK BY` option (in `DO`, `FOR EACH`, or `REPEAT` statements).

Each lock table entry takes 64 bytes.

Note: Validations using **-L** can be run online as part of routine health checks. For more information, see PROUTIL Utility on page 629.

---

[20] Limited by available memory.

Lock Table Hash Size (-lkhash)

Operating system and syntax	UNIX / Windows	-lkhash *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**[21]
DBS	–	–	–	–

Use Lock Table Hash Size (`-lkhash`) to specify the size of the hash table that controls access to the lock table.

Progress Software Corporation recommends changing the value of Lock Table Hash Size (`-lkhash`) only after contacting Technical Support.

Lock release (-lkrela)

Operating system and syntax	UNIX Windows	-lkrela		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Client Session	–	–	–	–

Lock Release (`-lkrela`) uses the original lock release mechanism installed from previous OpenEdge versions.

Using this parameter instructs the database to use the original lock release mechanism installed in previous versions of OpenEdge (versions 10.1A and higher).

Note: When using this parameter, keep in mind that the database engine can search all lock chains to be released at transaction end and disconnect time. As a result, you may see a degradation in performance.

LRU force skips (-lruskips)

Operating system and syntax	UNIX Windows	-lruskips *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	2147483648	0	–	0

n

The number of times to access a buffer in the Primary Buffer Pool before moving it to the end of the LRU chain.

[21] **The default is variable based on the size of your lock table (-l).**

Use LRU force skips (`-lruskips`) to specify the number of times a buffer in the buffer pool is accessed before it is placed on the Most Recently Used (MRU) end of the LRU chain. Specifying the LRU force skip parameter is helpful when high LRU latch wait timeouts are occurring (observable in PROMON or by VST). Accessing a buffer a specified number of times eliminates the need to acquire the LRU latch each time a buffer is accessed. When the LRU skip value is tuned to your environment, contention on the LRU latch decreases, improving concurrency and performance, and there is no increase in the number of page misses in the buffer pool. If increasing the LRU skip value causes increased buffer pool misses, then the value should be tuned down or turned off altogether. Setting `-lruskips` to zero disables the feature, and is the default.

Note: This parameter is only available for databases with an Enterprise license. LRU Force skips is not implemented for private buffer pool data access.

LRU alternate buffer pool force skips (-lru2skips)

Operating system and syntax	UNIX Windows	**-lru2skips *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	2147483648	0	–	0

n

The number of times to access a buffer in the Alternate Buffer Pool before moving it to the end of the LRU2 chain.

Use LRU alternate buffer pool force skips (`-lru2skips`) to specify the number of times a buffer in the alternate buffer pool is accessed before it is placed on the Most Recently Used (MRU) end of the LRU chain. Specifying the LRU alternate buffer pool skip parameter is helpful when high LRU latch wait timeouts are occurring (observable in PROMON or by VST) in the alternate buffer pool. Accessing a buffer a specified number of times eliminates the need to acquire the LRU latch each time a buffer is accessed. When the LRU alternate buffer pool skip value is tuned to your environment, contention on the LRU latch decreases, improving concurrency and performance, and there is no increase in the number of page misses in the buffer pool. If increasing the LRU skip value causes increased buffer pool misses, then the value should be tuned down or turned off altogether. Setting `-lru2skips` to zero disables the feature, and is the default.

Note: This parameter is only available for databases with an Enterprise license. LRU force skips is not implemented for private buffer pool data access.

Auto Server (-m1)

Operating system and syntax	UNIX / Windows	**-m1**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

Use Auto Server (`-m1`) to start an auto server. The OpenEdge broker uses the auto server internally to start a remote user server. This is the default. You will never have to use this parameter directly.

Secondary Login Broker (-m3)

Operating system and syntax	UNIX / Windows	-m3		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

In a network environment where more than one broker is using the same protocol, use Secondary Login Broker (-m3) to start each secondary broker. The secondary broker logs in clients and starts remote user servers.

Maximum Clients per Server (-Ma)

Operating system and syntax	UNIX / Windows	-Ma *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	2048	1	–	5 users/server

n

The maximum number of remote users per database server. The default is the Maximum Number of Users (-n) parameter value, divided by the Maximum Number of Servers (-Mn) parameter value.

Use Maximum Clients per Server (-Ma) to specify the maximum number of remote users per database server. The Maximum Clients per Server (-Ma), Minimum Clients per Server (-Mi), and Maximum Servers (-Mn) startup parameters apply only to databases that are accessed from remote network nodes.

In most cases, the default behavior is desirable. Note that the default calculation is usually high because it assumes that all users are remote users, while the number specified with -n includes local users. If servers become overloaded with clients, reset the -Mn parameter to increase the number of servers.

Maximum area number (-maxAreas)

Operating system and syntax	UNIX Windows	-maxAreas *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32000	Variable[22]	32000	32000

n

The maximum area number.

[22] The minimum value is the current maximum area number in use for the database.

Use Maximum area number (`-maxAreas`) to specify the highest area number available for use during the time the database is online. Increase the highest area number by shutting down and restarting the database with a higher value, or remove the `-maxAreas` parameter to have the maximum number available.

At database startup, the maximum area number specified with `-maxAreas` is compared to the areas defined in the database. If any defined area has a higher area number, the database will not start.

If the database is expanded with PROSTRCT ADDONLINE, the area numbers specified for any new areas must be lower than the maximum area number specified with `-maxAreas`, or the PROSTRCT ADDONLINE command will fail.

For more information on area numbers, see Maintaining Database Structure on page 325

Maximum Dynamic Server (-maxport)

Operating system and syntax	UNIX / Windows	**-maxport *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	System dependent	System dependent	–	2000

n

The port number that is the highest in a specified range.

Use Maximum Dynamic Server (`-maxport`) to specify the highest port number in a specified range of port numbers accessible to a client. You specify the lowest port number with the `-minport` parameter. The range of port numbers defined by the `-maxport` and `-minport` parameters provides client access to an OpenEdge server that is behind a firewall. Some operating systems choose transient client ports in the 32,768-to-65,535 range. Choosing a port in this range might produce unwanted results.

Maximum JTA Transactions (-maxxids)

Operating system and syntax	UNIX / Windows	**-maxxids *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32,000	100	–	100

n

The maximum number of simultaneous JTA transactions allowed.

JTA transactions rely on an external transaction manager to control transaction commit and rollback. OpenEdge SQL supports the Java Transaction API (JTA) enabling the database to function as a resource manager as defined in the J2EE framework architecture. Use the Maximum JTA Transactions parameter (`-maxxids`) to control the number of JTA transactions simultaneously allowed.

Memory consistency check (-MemCheck)

Operating system and syntax	UNIX / Windows	-MemCheck		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	–	–	–	–

Use Memory consistency check to enable memory overwrite checking for the buffer pool. Memory consistency check detects illegal operations in memory movement, such as a buffer copy with a negative length of data, or a memory move to an incorrect location. The purpose Memory consistency check is to prevent memory corruption.

Index and record block operations are checked prior to before-image note writing and the subsequent database block write. The length of the intended operation is checked to prevent operations that exceed database the block size.

Index block splits and index insert operations are checked at strategic points to monitor consistency.

Memory consistency checking can be enabled or disabled while your database is online with PROMON. See Enable/disable block level consistency check on page 618 for more information.

Memory consistency check can be enabled for a single user client or for any offline utility. If the utility is read-only, such as DBANALYS, the `-MemCheck` parameter is ignored. Online utilities determine whether or not to perform memory consistency checking during execution based on the use of `-MemCheck` at broker startup or by the enablement/disablement of the memory consistency checking in PROMON.

Delayed BI File Write (-Mf)

Operating system and syntax	UNIX / Windows	-Mf *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**[23]	**Multi-user default**
DBS	32,768	0	0	3

n

Value in seconds of the delay before the database engine synchronously writes to disk the last before-image file records at the end of each transaction. It also specifies the interval that the broker process wakes up to make sure all BI file changes have been written to disk. The default is 3 for single-user batch jobs and for multi-user databases. Otherwise, the default is zero (0).

Use Delayed BI File Write (`-Mf`) to improve performance on a heavily loaded system. Using `-Mf` does not reduce database integrity. However, if there is a system failure, it is possible the last few completed transactions will be lost (never actually written to the BI file).

[23] **Default is 3 for batch jobs.**

When running with full integrity, at the end of each transaction the database engine does a synchronous write to disk of the last BI file block. This write guarantees that the completed transaction is recorded permanently in the database. If the user is notified that the transaction has completed and the system or database manager crashes shortly afterwards, the transaction is not lost.

Do not set `-Mf` on a lightly loaded system with little database update activity. Under these conditions, the extra BI write is very important and does not impact performance. On a heavily loaded system, however, the BI write is less important (the BI block will be written to disk very soon anyway), and has a significant performance penalty. Setting `-Mf` to delay this extra BI write saves one write operation per transaction, which can significantly improve performance. The extra BI file write is delayed by default for batch jobs.

The last BI file record is only guaranteed to be written out to disk when a user logs out, or when the server or broker process terminates normally. On multi-user systems, the *n* argument determines the maximum length of time in seconds during which a completed transaction can be lost.

Minimum Clients per Server (-Mi)

Operating system and syntax	UNIX / Windows	**-Mi *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	1	–	1

n

The number of remote users on a server before the broker starts another server. See Maximum Servers (-Mn) on page 514 for more information.

Use Minimum Clients per Server (`-Mi`) to specify the number of remote users on a server before the broker starts another server (up to the maximum number of servers). In addition, `-Mi` and `-Mn` apply only to databases that are accessed from remote network nodes.

As remote users enter the database, the broker process starts just one server process for each *n* remote users, until the maximum number of servers (specified by the `-Mn` parameter) is started. If you specify a value of 1, the broker starts a new server for each of the first `-Mn` remote users. Subsequent remote users are distributed evenly among the servers until the maximum number of users (`-n`) or maximum clients per server (`-Ma`) limits are reached.

Typically, you can leave `-Mi` and `-Mn` at their default values. If you significantly increase `-Mn`, you should also increase `-Mi`. For example, if you set `-Mn` to 10 to accommodate up to 40 or more remote users, increase `-Mi` to 3 or 4 to prevent a situation where 10 servers are started for just 10 remote users.

Minimum Dynamic Server (-minport)

Operating system and syntax	UNIX / Windows	**-minport *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	1,025	–	1,025

n

The port number that is the lower in a specified range.

Use Minimum Dynamic Server (`-minport`) to specify the lowest port number in a specified range of port numbers accessible to a client. You specify the higher port number with the `-maxport` parameter. Ports below 1025 are usually reserved for system TCP and UDP. The range of port numbers defined by the `-maxport` and `-minport` parameters provides client access to an OpenEdge server that is behind a firewall. This communication is possible only when access to the server can be limited.

Message Buffer Size (-Mm)

Operating system and syntax	UNIX Windows	**-Mm *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	32,600	350	—	1,024

n

The message buffer size.

Use Message Buffer Size (`-Mm`) to specify the standard message buffer size, in bytes. This parameter is relevant only for network client/server connections. Message Buffer Size (`-Mm`) can be specified by both the client and the server, however the size of the message buffer specified by the server takes precedence; a value specified by the client is used as a suggestion for initial buffer allocation, but the client then adopts the server value when the connection is initiated.

OpenEdge uses message buffers to move records (messages) between servers and remote clients. Records (plus 40-byte headers) larger than the message buffer size are fragmented into multiple messages. If your database records are large, increase this parameter to avoid record fragmentation. However, if the network works more efficiently with small messages, reduce `-Mm` and fragment larger records.

Note: Prior to OpenEdge Release 11.6, the value of `-Mm` had to agree between the client and server or the connection would be refused. Going forward, in addition to not requiring agreement between client and server, clients can also connect to multiple databases with different `-Mm` values obtained from the server. To enforce the earlier behavior, use Enforce Mm (`-enforceMm`).

Maximum Servers (-Mn)

Operating system and syntax	UNIX / Windows	**-Mn *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	1	–	4

n

The maximum number of remote client servers that can be started on the system. The value specified is always incremented by 1 to provide a server for the primary login broker.

Use Maximum Servers (`-Mn`) to limit the number of remote user servers that can be started by the broker process. The performance tradeoff to consider is swapping overhead for many servers versus overloading (slowing down) a server with too many clients. Also, use Minimum Clients per Server (`-Mi`) to adjust the actual number of servers in use.

Note: The maximum value for the number of servers that can be started for a database is limited by available resources of the operating system.

See Maximum Clients per Server (-Ma) on page 510 and Minimum Clients per Server (-Mi) on page 513 for more information.

Servers per Protocol (-Mp)

Operating system and syntax	UNIX / Windows	**-Mp *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	Value of -Mn	1	–	value of -Mn

n

The number of servers a broker can start.

Use Servers Per Protocol (`-Mp`) with Secondary Login Broker (`-m3`) in database networks that use more than one network protocol. This parameter limits the number of servers that the broker can start to serve remote users for any one protocol. The total number of servers for all protocols is still limited by the Maximum Servers (`-Mn`) parameter.

Maximum Servers per Broker (-Mpb)

Operating system and syntax	UNIX / Windows	**-Mpb *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

n

The number of servers each broker can start.

Use Maximum Server Per Broker (`-Mpb`) to specify the maximum number of servers that multiple brokers can start to serve remote users for any one protocol.

User MUX Latches (-mux)

Operating system and syntax	UNIX / Windows	-mux *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBSCC	1	0	–	1

Use User MUX Latches (`-mux`) to control the granularity of access to large database resources in shared memory.

Do not use this parameter unless directed to do so by Progress Software Technical Support.

Shared-memory Overflow Size (-Mxs)

Use Shared-memory Overflow (`-Mxs`) to replace the default value of the shared-memory overflow area; it does not increase it.

Operating system and syntax	UNIX / Windows	-Mxs *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Database Server	System dependent[24]	1	–	For 32-bit platforms: 16384 + (*# of users* * 300) / 1024 For 64-bit platforms: 16384 + (*# of users* * 400) / 1024

n

The size of the shared-memory overflow area in kilobytes.

The overflow area is appended to the shared-memory area. If the overflow area is too small, the database engine exits with the following message:

```
SYSTEM ERROR: Out of free shared memory. Use -Mxs to increase.
```

Depending on the operating system, the database engine rounds the shared-memory area size to the next 512-byte or 4K boundary.

Note: Validations using **-Mxs** can be run online as part of routine health checks. For more information, see PROUTIL Utility on page 629.

[24] The maximum is limited only by the size of the signed integer data type on the system.

Specific memory requirements when using mandatory fields

If you are using mandatory fields, consider the following approximate memory requirements when calculating the value of Shared Memory Overflow (-Mxs):

- 480 byte overhead
- 48 bytes for each table containing mandatory fields
- 12 bytes for each mandatory field

When calculating Shared Memory Overflow (-Mxs), add the total number of bytes to the existing default value. For example, if the total memory required for the mandatory field array is 80K, -Mxs must be (80K + 16K + 4K (100 users).

Note: These numbers are approximations, and may change from release to release.

Network Type (-N)

Use Network Type (-N) to specify the network communications protocol supported by the OpenEdge RDBMS.

Operating system and syntax	UNIX / Windows	-N *network-type*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Client Connection, Database Server	–	–	–	System dependent

network-type

The network communications protocol. TCP is the only supported option.

Number of Users (-n)

Operating system and syntax	UNIX / Windows	-n *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	10,000	–	–	20

n

The maximum number of OpenEdge users on the system. After *n* users have connected to the OpenEdge database, additional user startup attempts are rejected.

-n must be high enough to include local and remote users as well as background writers (APWs, BIWs, and AIWs), PROWDOG processes, and PROMON sessions.

See Maximum Clients per Server (-Ma) on page 510 and Maximum Servers (-Mn) on page 514 for more information.

Nap Time Steps (-napstep)

Operating system and syntax	UNIX / Windows	-napstep *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	5

Use Nap Time Steps (`-napstep`) to specify the steps between the nap increments for Enterprise database licenses only.

Do not use this parameter unless directed to do so by Progress Software Technical Support.

Login Governor (-nGovernor)

Operating system and syntax	UNIX Windows	-nGovernor *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Database Server	32000	1	–	–

n

The maximum number of users.

Login Governor (`-nGovernor`) specifies the maximum number of users that may login for each tenant in a multi-tenant database. Login governor prevents a tenant from using all the database connections, and prevents other tenants from logging in. Login governor applies to the default tenant and regular tenant. This governor does not apply to super-tenants and non multi-tenant databases.

When a user attempts to log into a database, and the available login resource allocation for the tenant has been exceeded, an error message is recorded in the database log as:

```
Attempt to exceed tenant login governor for tenant <tenantID>. Increase -n or -nGovernor.
```

You can set or modify the `-nGovernor` value using the PROMON utility. For more information about the PROMON utility, see PROMON Utility on page 543. To set the `-n` value, see Number of Users (-n) on page 517.

Message Wait (-Nmsgwait)

Operating system and syntax	UNIX Windows	-Nmsgwait *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**[25]
Database Server	255	1	–	2

n

The number of seconds a server waits for a remote network message.

Use Message Wait (`-Nmsgwait`) to indicate the number of seconds a server waits for a remote network message before checking for other events such as a database shutdown or forced disconnect of a remote client.

No Session Cache (-nosessioncache)

Operating system and syntax	UNIX Windows	-nosessioncache		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

Use No Session Cache (`-nosessioncache`) to disable SSL session caching.

Session caching allows a client to reuse a previously established session if it reconnects prior to the session cache time-out expiring. Use No Session Cache (`-nosessioncache`) to disable this feature. Session caching is enabled by default.

Number of checkpoint statistics (-numcheckpointstats)

Operating system and syntax	UNIX / Windows	-numcheckpointstats *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	1024	8	–	32

n

The number of checkpoints to track statistics for in the `_Checkpoint` VST.

Use Number of checkpoint statistics (-numcheckpointstats) to specify the number of rows maintained in the`_Checkpoint` VST.

[25] **The default of 2 produces pre-Release 11.1 behavior.**

Storage Object Cache Size (-omsize)

Operating system and syntax	UNIX / Windows	-omsize *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	65535	0	–	1024

Use Storage Object Cache Size (`-omsize`) to specify the size of the object cache for all database objects.

Prompt for Passphrase (-Passphrase)

Operating system and syntax	UNIX / Windows	-Passphrase		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBSCC	–	–	–	–

Use Prompt for Passphrase (`-Passphrase`) to open an encryption-enabled database that is configured for manual start, or to override the autostart passphrase configuration.

Database utilities and servers indicate that they require a prompt for key store authentication by adding Prompt for Passphrase (`-Passphrase`) to the command line. Prompt for Passphrase prompts for the passphrase. If you cannot successfully authenticate the key store, you will not be able to open to the database.

For more information on Key Store Passphrase, and Transparent Data Encryption in general, see Transparent Data Encryption on page 199.

Pending Connection Time (-PendConnTime)

Operating system and syntax	UNIX / Windows	-PendConnTime *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	0

n

Time, in seconds, allowed the client to connect to a server.

Use Pending Connection Time (`-PendConnTime`) to prevent connection errors caused by network problems.

When a client attempts to connect to an OpenEdge database, the RDBMS assumes that if the client can reach the broker, it can also reach the server. Network problems or incorrect configuration of network devices can prevent the client from reaching the server. In such an instance, the broker, unaware that the client's connection failed, continues to increment its count of connected users. To prevent this problem, OpenEdge brokers use a "reservation" count on each server and they increment this count whenever they redirect a client to that server.

When `-PendConnTime` is used, the servers examine the timestamp on the latest reservation and, if the Pending Connection Time period has elapsed, the broker assumes that the client has failed to connect. The broker then clears the reservation, making the server available for new clients.

Parameter File (-pf)

Operating system and syntax	UNIX / Windows	**-pf *filename***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, CS, DBS, DS	–	–	–	–

filename

The name of the parameter file containing OpenEdge database startup parameters.

Use Parameter File (`-pf`) to name a parameter file that includes any number of startup parameters to run an OpenEdge database. This parameter is especially useful if you regularly use the same parameters to start your database, or if more parameters are specified than can fit on the command line. This parameter can be included within the parameter file itself to reference another parameter file.

Use multiple instances of `-pf` to name multiple parameter files. This allows you to specify application-specific parameters in one parameter file, database-specific parameters in a second parameter file, and user-specific parameters in yet another file.

Database Service Communication Area Size (-pica)

Operating system and syntax	UNIX / Windows	**-pica *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	1000000	4	–	16

Use Database Service Communication Area Size (`-pica`) to set the size of the database service communications area in Kb. The database service communications area is used to store after-image block write notifications intended for OpenEdge Replication.

Pin Shared Memory (-pinshm)

Operating system and syntax	UNIX	-pinshm		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

The Pin Shared Memory (`-pinshm`) parameter does not have any arguments.

Using `-pinshm` will prevent the OS from swapping shared memory contents to disk, which can help improve performance. Pin Shared Memory (`-pinshm`) is only supported on UNIX platforms, excluding IBM AIX.

Prefetch delay (-prefetchDelay)

Operating system and syntax	UNIX Windows	-prefetchDelay		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

Use Prefetch delay (`-prefetchDelay`) to enable a delay when sending the first network message for queries with prefetch capabilities.

When you specify Prefetch delay, Prefetch Num Recs (-prefetchNumRecs) on page 523 and Prefetch factor (-prefetchFactor) on page 522 parameters determine the duration of the delay. Prefetch delay is off by default, indicating the that first network message is sent as soon as it contains the first record of the query.

Prefetch factor (-prefetchFactor)

Operating system and syntax	UNIX Windows	-prefetchFactor *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**[26]
DBS	100	0	–	0

n

The percent of a message buffer to fill before sending it to a waiting remote client.

Use Prefetch Factor (`-prefetchFactor`) to establish a percentage of a network message required to contain prefetched data before sending the message to a waiting remote client.

The number of records in a network buffer is also influenced by Prefetch Num Recs (-prefetchNumRecs) on page 523 parameter. See Prefetch tuning on page 306 for a discussion of the interaction between the parameters.

[26] **The default of 0 gives pre-Release 11.1 behavior**

Prefetch Num Recs (-prefetchNumRecs)

Operating system and syntax	UNIX Windows	-prefetchNumRecs *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**[27]
DBS	32766	1	–	16

n

The number of prefetch records to put in a network message before sending the message to a waiting remote client.

Use Prefetch Num Recs (`-prefetchNumRecs`) to increase the number of records placed in a network message before it is sent to a remote client. Setting Prefetch Num Recs (`-prefetchNumRecs`) may cause a waiting remote client to wait longer for full network messages containing prefetched records, but overall system performance is expected to improve by saving on network traffic. Another possible side-effect is that an individual remote client see might see "choppy" responses from the server.

The number of records in a network buffer is also influenced by Prefetch factor (-prefetchFactor) on page 522 parameter. See Prefetch tuning on page 306 for a discussion of the interaction between the parameters.

Prefetch Priority (-prefetchPriority)

Operating system and syntax	UNIX Windows	-prefetchPriority *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**[28]
DBS	32766	0	–	0

n

The number of prefetch records to add to a network message.

Use Prefetch Priority (`-preFetchPriority`) to utilize a "pollskip" that adds *n* records to the network message of an in-process prefetch query without additional polling. Adding more than one record to the message at a time can greatly reduce the number of `poll()` calls being made on a large database configuration. The expected result is that reducing `poll()` calls improves overall system performance by decreasing the "System CPU" cycles allow for increased "User CPU" cycles. The pollskip only occurs if the previous `poll()` call returned nothing waiting.

27 **Prior to Release 11.1, 16 was the static setting.**

28 **The default of 0 produces the pre-Release 11.1 behavior.**

Configuration Properties File (-properties)

Operating system and syntax	UNIX / Windows	-properties *filename*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

filename

The name of the properties file.

Use Configuration Properties File (-properties) to identify the properties file that the AdminServer uses internally to specify startup parameters when starting a database server or server group. The default is $DLC/properties/conmgr.properties. You do not use this parameter directly.

Buffered I/O (-r)

Operating system and syntax	UNIX / Windows	-r		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC	–	–	Unbuffered I/O	Unbuffered I/O

Use Buffered I/O (-r) to enable buffered I/O to the before-image file. In most cases, avoid using this parameter because it might put database integrity at risk.

Caution: A database running with the -r parameter cannot be recovered after a system failure. If the system fails, you must restore the database from a backup and restart processing from the beginning.

Record free chain search depth factor (-recspacesearchdepth)

Operating system and syntax	UNIX / Windows	-recspacesearchdepth *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	100	0	0	5

n

The percentage of the cluster size.

Use Record free chain search depth factor (-recspacesearchdepth) to specify how far into the record free chain the database engine searches for a block with enough free space to store the data associated with a record create or update before a new cluster of data is allocated. The value is a percentage of the cluster size with a minimum of 3 blocks.

Service Name (-S)

Operating system and syntax	UNIX / Windows	-S { *service-name* | *port-number* }		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	–	–	–	–

service-name

The name of the service to be used by the broker process.

port-number

The port number of the host; if using OpenEdge Management or OpenEdge Explorer, the port number of the NameServer.

Use Service Name (-S) to specify the service or port number to be used when connecting to a broker process or used by a broker process on the host machine. You must use this parameter when you are starting:

- A broker or server on a machine that will serve remote users
- A multi-user session as a remote user

The system administrator must make an entry in the services file that specifies the server or broker name and port number.

When the broker spawns a server, the server inherits all of the network parameters (except the Service Name parameter) from the broker. Because there is no restriction on the number of brokers you can start, you can have multiple brokers running with different network parameters. See Server Group (-servergroup) on page 528 for more information.

The table below shows how the broker, server, and remote client interpret each of their parameters when you use the -S parameter.

Table 77: Parameter interpretation with service name (-S)

Module	Interpretation
Broker	Parameters apply to the connections on which the broker is listening for connection requests from remote clients
Server	Parameters apply to the connection between the server and the remote client
Remote Client	Parameters identify the connection parameters to the broker or the server

To run a multi-user OpenEdge session from a remote network node, use both the Host Name (-H) and Service Name (-S) parameters.

Schema Lock Wait Queue(-schlockwq)

Operating system and syntax	UNIX / Windows	-schlockwq		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	3

Schema Lock Wait Queue (`-schlockwq`) alters the schema locking algorithm.

When the database broker is started with `-schlockwq` specified, the schema lock scheduler forces a shared schema lock request to wait when an exclusive schema lock request is queued. Once an exclusive schema lock request is queued, it appears to all subsequent schema lock requests that an exclusive schema lock has been granted (meaning all subsequent schema lock requests will queue). Without `-schlockwq`, shared locks continue to be granted, even though an exclusive lock is queued.

With `-schlockwq`, wake up of users waiting for a schema lock is performed in the order they were received. Without `-schlockwq`, all users are awoken whenever any schema lock conflict is resolved, causing a race to see which user gets the available lock. When running with `-schlockwq` enabled at broker startup, only those users allowed to get the schema lock will be awoken.

All existing lock strength conflicts and properties are unchanged, as follows:

- Exclusive locks conflict with shared and exclusive locks for other users.
- Share locks may coexist with other share locks but conflict with exclusive locks of other users.
- An individual user may possess at most one shared lock and one exclusive lock concurrently.
- An individual user may never possess more than one lock of the same lock strength (i.e. two concurrent share locks for the same user is prohibited).
- An individual user may only ever have one entry on the wait queue.
- These behaviors are inherently supported by each of the language engines and enforced by the storage engine.

The `-schlockwq` parameter has no impact when the database is started in single-user mode.

Security cache size (-secsize)

Operating system and syntax	UNIX / Windows	-secsize *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32768	0	-	512

n

The size, in number of cache entries, of the primary and secondary security caches.

For best performance, you should set Security cache size (-secsize) large enough that all your existing records fit in the primary cache. Pick the value for -secsize based on available shared memory and the size of the _sec-authentication-domain table. Each entry consumes approximately 450 bytes. For example, if -secsize is 512, roughly 512 * 450 = 0.25 Mb is allocated for the primary cache, and also the secondary cache, for a total of 0.5 Mb. If memory is not an issue, pick a value close the number of records in _sec-authentication-domain.

If you set the value between 1 and 12, it is automatically increased to 13.

You can set Security cache size (-secsize) to 0 (zero) if you never intend to enable use of the security cache on your database.

For use with the OpenEdge Authentication Gateway only, use Security Cache Size (-secsize) to specify the size (in number of security cache entries) of the security cache. The security cache holds fields from the _sec-authentication-domain and _Tenant-Name tables in memory, eliminating the need to read those values from the database during connection validation.

At startup, if your database is currently enabled for authentication through use of the OpenEdge Authentication Gateway, two security caches are allocated, a primary cache and a secondary cache.

Existing fields are cached in the primary security cache and are preloaded by a broker. After instantiation, no new entries are ever added to the primary cache. If the primary cache is filled, remaining entries are inserted into the secondary cache. All new entries are automatically cached in the secondary cache. During normal database activity, fifty percent (50%) of the cache entries must be evictable. If the number of fixed cache entries is more than 50% of all entries, new entries cannot be added to the cache and creation of a new _sec-authentication-domain record fails. If this occurs, an informational message is written to the database log file, as shown:

```
Please increase security cache size using database startup parameter -secsize.
Cannot perform this operations because the database security cache is too small.
```

Semaphore Sets (-semsets)

Operating system and syntax	UNIX	-semsets *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	Maximum Number of Users + 1	1	–	3

n

An integer specifying the number of semaphore sets available to the broker.

Use Semaphore Sets (-semsets) to change the number of semaphore sets available to the broker. When more than 1,000 users connect to a single database, there might be high contention for the semaphore set. If there is a lot of semaphore contention on a system, using multiple semaphore sets helps improve performance on high user counts.

Server Group (-servergroup)

Operating system and syntax	UNIX / Windows	-servergroup *name*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

name

Specifies the name of the logical collection of server processes.

Use Server Group (`-servergroup`) to identify the logical collection of server processes to start. The *name* you specify must match the name of a server group in the `conmgr.properties` file. Use OpenEdge Management or OpenEdge Explorer to create server groups and save them in the `conmgr.properties` file. Do not edit the `conmgr.properties` file directly. To start a database configuration, use OpenEdge Management, OpenEdge Explorer, or the DBMAN utility.

Session Timeout (-sessiontimeout)

Operating system and syntax	UNIX Windows	-sessiontimeout *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	180

n

Specifies in seconds the length of time an SSL session will be held in the session cache. The default is 180 seconds.

Use Session Timeout (`-sessiontimeout`) to change the length of time that an SSL session will be cached. Session caching allows a client to reuse a previously established SSL session if it reconnects prior to the session cache timeout expiring.

Shared memory segment size (-shmsegsize)

Operating system and syntax	UNIX Windows	`-shmsegsize n`		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	System dependent[29]	128 for 32-bit systems, 1024 for 64-bit systems.	–	Configuration and system dependent.[30]

n

The maximum number of bytes, specified in MB or GB, in a shared memory segment. The table below lists the supported values by platform.

Table 78: Values for -shmsegsize

Platform	Values in MB	Values in GB[31]
32-bit platforms	128	–
	256	–
	512	–
	1024	1g
	2048	2g
	4096	4g
64-bit platforms	1024	1g
	2048	2g
	4096	4g
	8192	8g
	16384	16g
	32768	32g

30 If not specified, the first attempt is to allocate one segment for all the required memory. If the required memory exceeds the operating system maximum segment size, the system maximum size is used.

29 Limited by available memory.

31 You must include the "g" if you specify the value in gigabytes.

Use Shared memory segment size (`-shmsegsize`) to specify the size of the largest shared memory segment the server can allocate. Specifying shared memory segment size can improve performance. Increasing the size of the shared memory segments decreases the number of segments allocated, in turn decreasing the system resources needed to manage the segments. See Shared memory allocation on page 297 for more information.

Notes

- If `-shmsegsize` is omitted, OpenEdge attempts to fit all shared memory into the fewest segments possible.
- If the total shared memory required is smaller than the value specified with `-shmsegsize`, only the required amount is allocated.

Spin Lock Retries (-spin)

Operating system and syntax	UNIX / Windows	`-spin` *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	0	–	10,000, or 6000 * # CPUs[32]

n

The number of times a process tries to acquire a latch before pausing.

Use Spin Lock Retries (`-spin`) to set a value to use the spin lock mechanism or a system of semaphores and queues. If the value of *n* is greater than zero (0), a spin lock algorithm is used for shared-memory data structures. When a process has to lock a shared-memory structure, the process tries up to *n* times to acquire the latch for that structure. If it has not acquired the latch in *n* tries, then the process pauses, or naps. The length of the pause increases gradually if the process repeatedly fails to acquire a latch. After the allotted nap time, the process wakes up and tries to acquire the lock again. If it fails to acquire the lock, it will retry up to the number of tries specified by *n*.

If the value of *n* is zero (0), a system of semaphores and queues is used instead of spin locks. The spin lock algorithm is much more efficient than using semaphores when you have multiple processors.

On multi-processor machines, the default value is 6000 * # of CPUs. If this causes too much CPU usage, reduce the value. If you have many fast processors, a value as high as 10,000 * # of CPUs might be effective.

You can evaluate the `-spin` values through the PROMON utility's **R&D** options. See PROMON R&D Advanced Options on page 565. The `-spin` parameter controls the performance indicator called resource waits. By setting the `-spin` value higher, you can reduce the resource waits. Note that when setting the `-spin` value higher ceases to reduce the resource waits, continuing to set it higher can adversely effect CPU utilization. To view the resource waits value:

- Access PROMON and enter `R&D` at the main menu.
- Choose `option 3, Other Displays`, then choose `option 1, Performance Indicators`, to view the resource waits. Resource waits is the last item reported in the listing.

[32] For single CPU systems, the default is 10,000. For multi-CPU systems, the default is 6000 * # of CPUs.

Lock Wait Timeout (-SQLLockwaitTimeout)

Operating system and syntax	UNIX / Windows	**-SQLLockwaitTimeout *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	—	—	5	5

n

Use this parameter to specify the number of seconds to wait when a lock conflict occurs. This value applies to all conflicts that occur while using SQL applications. The default value is 5 seconds.

Note: Progress Software Corporation recommends changing the value of Lock Wait Timeout(-SQLLockwaitTimeout) to a higher value if your installation encounters a lot of conflicts.

SQL rowid identifier(-SQLQuotedRowid)

Operating system and syntax	UNIX Windows	**-SQLQuotedRowid [std | *value*]**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	std	std

std | *value*

Specify behavior of the string "rowid":

- std — specify std to indicate that all instances of the quoted string "rowid" are interpreted as an SQL identifier (such as a column name).
- *value* — specify any string other than std to indicate that all instances of rowid, quoted or not, are interpreted as the SQL rowid function.

Use SQL rowid identifier (-SQLQuotedRowid) to specify how the quoted string "rowid" is interpreted.

Example

Start the database server as shown:

```
mprosrv myDB -S  1234 -H myhost  -SQLQuotedRowid std
```

After starting the server and connecting a client to the database, query a column named rowid as shown:

```
select custnum, "rowid"  from pub.my_customer where "rowid" = 10 ;
```

Query both the column named rowid (quoted), and the value of the rowid function(no quotes) as shown:

```
select custnum, "rowid", rowid from pub.my_customer where "rowid" = 10 ;
```

Authorized Data Truncation(-SQLTruncateTooLarge)

Operating system and syntax	UNIX / Windows	-SQLTruncateTooLarge { ON | ALL | OFF | OUTPUT }		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	—	—	OFF	OFF

OUTPUT

Set this parameter to OUTPUT to truncate the data returned to the user when its size exceeds the column size. All SQL operations such as comparison, sorting use actual data and only the output is truncated.

ON/ALL

Set this parameter to ON or ALL to perform SQL operations on truncated data. Also the data returned to the user is also truncated when its size exceeds the column size.

OFF

When the parameter value is set to OFF, data will not be truncated and an error is returned.

Autonomous Schema Update (-SQLWidthUpdate)

Operating system and syntax	UNIX / Windows	-SQLWidthUpdate *ON*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	—	—	OFF	OFF

ON

Set the value of this parameter as *ON*, during startup, to resolve the SQL width problem by updating the SQL column width in schema automatically when Authorized Data Truncation(ADT) occurs for data in a column. Once the value of the parameter is set, it is remembered for the lifetime of the server and for all connections. If this parameter is set to *ON* during startup, the Authorized Data Truncation(_SQLTruncateTooLarge) parameter is also enabled irrespective of whether ADT was enabled during startup unless specified otherwise using connection URL.

Autonomous Update Statistics (-SQLAutoStats)

Operating system and syntax	UNIX / Windows	**-SQLAutoStats *ON***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	—	—	OFF	OFF

ON

Set the value of this parameter as *ON*, during startup, to start the Autonomous Update Statistics (AUS) service. Once the value of the parameter is set, it is remembered for the lifetime of the server and for all connections.

Server-side JOIN (-ssj)

Use Server-side JOIN (-ssj) to indicate if the database server will process Server-side JOIN requests from remote clients.

Operating system and syntax	UNIX / Windows	**-ssj *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Database Server	1	0	—	0 — for non-threaded server 1 — for threaded server

n

Set *n* as follows to indicate the status of Server-side JOIN processing:

- **0** — Server-side JOIN processing is disabled for this server. This is the default for a non-threaded database server.
- **1** — Server-side JOIN processing is enabled for this server. This is the default for a threaded database server.

Server-side JOIN is only enabled for certain queries if the tables to be joined are in the same logical database and there are ten or fewer tables. For 12.0 Server-side JOIN is limited to queries that use NO-LOCK.

The -ssj parameter is ignored if it's included in a server startup command with -threadedServer 0.

SSL (-ssl)

Operating system and syntax	UNIX Windows	-ssl		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

Use SSL (`-ssl`) to specify that all database and client connections use the Secure Sockets Layer (SSL) for data privacy. SSL provides an authenticated and encrypted peer-to-peer TCP/IP connection.

Note: SSL incurs heavy performance penalties, depending on the client, server, and network resources and load. For more information on SSL and the security features of OpenEdge, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

STS logging level (-stsloggінglevel)

Operating system and syntax	UNIX / Windows	-stsloggінglevel *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS,CC	5	0	0	0

n

The logging level:

- **0** — none
- **1** — fatal errors only
- **2** — all errors
- **3** — all errors and warnings
- **4** — all errors, warnings, and informational messages
- **5** — all errors, warnings, informational, and debug messages
- Values less than 0 generate an error; values greater than 5 are reset to 5 internally

For use with the OpenEdge Authentication Gateway only, use STS logging level (`-stsloggінglevel`) to specify the logging level value for STS client interface logging. This client-side startup parameter can aid you in debugging authentication problems in an OpenEdge Authentication Gateway deployment.

Notes

- This is a per database connection setting. If you connect to more than one database, each database can have a different `-stsloggінglevel` value.
- All output generated by `-stsloggінglevel` is written to the database `.lg` file.

- If the primary broker specifies `-stslogginglevel` at startup, the shared default is changed to the broker's value. Automatic servers and secondary brokers can only use the shared default. This parameter is not listed as a database startup parameter in the `.lg` file, VSTs or promon.
- Automatic servers and secondary brokers can only use the shared default value.
- Single user connections, self service clients and manual servers use the default shared value by default, but can override the use of the default by specifying their own `-stslogginglevel` value upon connection.
- Remote clients use their database servers' setting according to the above rules.

Temporary Directory (-T)

Operating system and syntax	UNIX / Windows	**-T *dirname***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS,DBS	—	—	Working directory	Working directory

dirname

The directory in which to create temporary files.

Use Temporary Directory (`-T`) to specify a directory for temporary files.

If you do not use this parameter, OpenEdge creates temporary files in the current working directory.

The database broker passes the setting of `-T` to any servers it spawns. Manual servers and secondary brokers can specify their own `-T` or take the default; they do not inherit the setting from the primary broker. Remote Client connections can also specify a `-T` setting different from the server it is connecting to. Temporary space needed in the remote client is stored in its own `-T` directory. Temporary space needed in the server to process a request for a remote client is stored in the server's `-T` directory.

On UNIX and in Windows, these filenames begin with one of the prefixes listed in the table below, and are followed by a number.

Note: Access the `-T` parameter using the `SESSION:TEMP-DIR` attribute. This attribute is read-only.

The following table shows descriptions of these prefixes:

Temp file	Usage
`bcf`	Internal file for tracking recid changes
`dbg`	Debugger listing file
`dbi`	Stores temporary tables
`lbi`	Local before-image (subtransaction undo)
`lst`	Temporary file used by the compile-listing feature of the COMPILE statement
`obj`	Temporary `.r` file

Temp file	Usage
`*.ped`	Edit buffer contents. `ped` is NOT a prefix, it is a file extension.
`rcd`	Cache of r-code being run in a session
`rpf`	Raw data used by the Profiler
`srt`	Temporary sort space
`trp`	Stores Data Dictionary changes until they are saved

On UNIX, OpenEdge names these files uniquely for each user to avoid filename conflicts. Furthermore, OpenEdge stores these files as "unlinked" with no visible name in the UNIX file system, unless you use Save Temp Files (`-t`).

In Windows, if two or more users share the same working directory and there is a conflict of temporary files, the following message appears:

```
Unable to open or create filename, error 3
```

Table consistency check (-TableCheck)

Operating system and syntax	UNIX / Windows	**-TableCheck** { ***tablename*** | ***ownername.tablename*** | ***ownername.tablename.partitionname*** }		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CC, DBS	–	–	–	–

tablename

Specifies the name of the table for consistency checking.

owner-name

Specifies the owner of the table you want to check. You must specify an owner name unless the table's name is unique within the database, or the table is owned by PUB. By default, ABL tables are owned by PUB.

partitionname

In a database with partitioned tables, specifies a named partition. Specify the composite initial partition by using `initial` as *partitionname*.

Use Table consistency check to enable consistency checking for all record blocks of the specified table (excluding BLOBS) or partition of a table. Table consistency check validates a block is still physically correct after an operation has been performed.

You can only specify one table or partition with `-TableCheck`. To check multiple partitions, apply consistency checking to the entire table. To check multiple tables, you must apply consistency checking to the entire database with Database consistency check (-DbCheck) on page 498 or if multiple tables are in one area, you can apply consistency checking to an entire area with Area consistency check (-AreaCheck) on page 485.

When specifying the table or partition to check, the following parsing rules are applied:

- If one variable is supplied, it is interpreted as a table name.
- If two variables are supplied, they are interpreted as the owner and table name (*ownername.tablename*).
- Three variables **must** be supplied to specify a table partition (*ownername.tablename.partitionname*).

Table consistency checking can be enabled or disabled while your database is online with PROMON. See Enable/disable block level consistency check on page 618 for more information.

Table consistency check can be enabled for a single user client or for any offline utility. If the utility is read-only, such as DBANALYS, the `-TableCheck` parameter is ignored. Online utilities determine whether or not to perform table consistency checking during execution based on the use of `-TableCheck` at broker startup or by the enablement/disablement of the table consistency checking in PROMON.

Table Range Size (-tablerangesize)

Operating system and syntax	UNIX / Windows	**-tablerangesize *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	–	–	–	–

n

The number of tables for which you want to track access statistics.

Use Table Range Size (`-tablerangesize`) to specify the number of tables for which you want to collect statistics.

Threaded Server (-threadedServer)

Use the Threaded Server (-threadedServer) parameter to indicate a server or servers spawned by this broker uses threads, one per remote client.

Operating system and syntax	UNIX / Windows	**-threadedServer [0 | 1]**		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Database Server	1	0	—	1

0

Indicates that servers spawned by the broker are not threaded.

1

Indicates that the servers spawned by the broker are threaded. This is the default.

The -threadedServer parameter is specified for each broker. When set to 1, servers spawned by this database broker service remote client requests using individual threads, one thread per remote connection.

The value of -threadedServer can not be changed at run time.

Threaded Server Stack (-threadedServerStack)

Use the Threaded Server Stack(-threadedServerStack) parameter to specify the size of a stack allocated for each thread spawned by a threaded server.

Operating system and syntax	UNIX / Windows	-threadedServerStack *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
Database Server	2147483647	64	—	512

n

Indicates the size, in Kb, of a stack reserved for each thread of a threaded server.

The -threadedServerStack parameter is specified for each broker, and can not be changed at run time.

Transaction End (TXE) Lock Retry Limit (-TXERetryLimit)

Operating system and syntax	UNIX / Windows	-TXERetryLimit *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	100,000	0	–	0

n

The number of tables for which you want to track access statistics.

Use Transaction End (TXE) Lock Retry Limit (-TXERetryLimit) to specify the number of number of times a connection will re-try to obtain TXE lock after a 1ms nap before getting queued. The default value is 0 (no behavior change).

The Transaction End (TXE) Lock is required for crash recovery to work properly. The lock single threads "micro-transaction" type updates within the storage engine, as well as assures that database changes are properly ordered when a transaction ends. A high degree of conflict queuing this lock can severely impact performance. The cost of managing the queue and waiting on a semaphore, as well as awakening waiting requests, while keeping things "fair" carries a high performance penalty.

When the connection spins on the TXERetryLimit, rather than waiting on a semaphore, contention shifts from the TXE Lock to the TXQ latch, but the lower level TXQ latch is a much lighter-weight mechanism, and because of that, improves the overall performance of the system. Additionally, the behavior of the TXQ latch conflicts can be further tuned using the existing parameters Spin Lock Retries (-spin) on page 530 and Nap Maximum (-napmax).

The best value for `-TXERetryLimit` is varied, but testing by Progress Software with a value of 200 showed the best performance improvement for the scenario where multiple users are creating and deleting large numbers of records in very small transactions, which previously had displayed a severe performance penalty.

User index range (-userindexrangesize)

Operating system and syntax	UNIX / Windows	**-userindexrangesize *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32765	1	50	50

n

The number of user indexes for which you want gather information in the `_UserIndexStat` VST.

Use User index range (`-userindexrangesize`) to specify the number of rows maintained in the`_UserIndexStat` VST.

Note: If the User index base and range are not specified, the `_UserIndexStat` VST is dimensioned using the values specified for the `_IndexStat` VST (`-baseindex` and `-indexrangesize`).

User table range (-usertablerangesize)

Operating system and syntax	UNIX / Windows	**-usertablerangesize *n***		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	32765	1	50	50

n

The number of user tables for which you want gather information in the `_UserTableStat` VST.

Use User table range (`-usertablerangesize`) to specify the number of rows maintained in the`_UserTableStat` VST.

Note: If the User table base and range are not specified, the `_UserTableStat` VST is dimensioned using the values specified for the `_TableStat` VST (`-basetable` and `-tablerangesize`).

Database Client Notification (-usernotifytime)

Operating system and syntax	UNIX / Windows	-usernotifytime *polltime*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
DBS	86400	0	–	0

polltime

The time, in seconds, of how often a client polls for notification of a schema change. The default is 0, indicating that no polling occurs. The minimum non-zero value is 1; the maximum is 86400 (for 24 hours).

Use Database Client Notification (`-usernotifytime`) to specify how often a client polls the database to see if a schema change is in process.

If no updates to schema are anticipated, due to index activation or rebuild, while the database is online, you can disable notification, or set the notification value to a very high number. If a change is required, the time between polls can be modified with PROMON or through the _DbParams VST. However, if the value is changed from 0 to a non-zero value, currently connected clients will not observe the change; only new clients will poll to receive the polltime value.

Century Year Offset (-yy)

Operating system and syntax	UNIX Windows	-yy *n*		
Use with	**Maximum value**	**Minimum value**	**Single-user default**	**Multi-user default**
CS, DBS	–	–	1950	1950

n

A four-digit year (1990, for example); the default is 1950.

Use Century Year Offset (`-yy`) to determine the start of the 100-year period in which a date with a two-digit year is valid.

Some OpenEdge applications reserve only two-digits for the year in the date format. For example, when `-yy` is set at 1950, the server determines if the two-digit year value is greater or less than 50. If the year is greater than 50, the server assumes that the date is in the twentieth century. If the year is less than 50, the server assumes that the date is in the twenty-first century.

Table 79: Century setting and date format

-yy	Year (as supplied in DATE format)	Result of year function
1900	50 to 99	1950 to 1999
	00 to 49	1900 to 1949

-yy	Year (as supplied in DATE format)	Result of year function
1950	50 to 99	1950 to 1999
	00 to 49	2000 to 2049
1980	80 to 99	1980 to 1999
	00 to 79	2000 to 2079

Notice that all two-digit year values expand into the 100-year period beginning with `-yy`.

To test the effect of `-yy`, start the database with a different `-yy` value and run the following procedure:

```
DEFINE VARIABLE dDate AS DATE NO-UNDO.
DISPLAY "Enter date or Press F4 to end." WITH FRAME msg.

REPEAT:
  SET ddate LABEL "Date" WITH SIDE-LABELS.
  DISPLAY YEAR(dDate) FORMAT "9999" LABEL "=> Year" WITH SIDE-LABELS.
END.
```

Note: If you use a hard-coded date containing a two-digit year in a `.p` file, the OpenEdge RDBMS honors the `-yy` parameter and expands the two-digit year to a four-digit year during compilation. However, this might not match the run time `-yy`. To prevent this problem, use four digit years for hard-coded dates in programs.

This startup parameter provides the same functionality as the `SESSION:YEAR-OFFSET` attribute.

24

PROMON Utility

This chapter describes the OpenEdge database administration utility PROMON.

For details, see the following topics:

- PROMON utility
- PROMON User Control option
- PROMON Locking and Waiting statistics option
- PROMON Block Access option
- PROMON Record Locking Table option
- PROMON Activity option
- PROMON Shared Resources option
- PROMON Database Status option
- PROMON Shut Down Database option
- PROMON Currently Connected Tenants option
- PROMON R&D Advanced Options
- R&D Status Displays
- R&D Activity Displays
- R&D Other Displays
- R&D Administrative Functions

- R&D Adjust Monitor Options
- PROMON 2PC Transactions Control option
- PROMON Resolve 2PC Limbo Transactions option
- PROMON 2PC Coordinator Information option
- PROMON Resolve JTA Transactions option
- PROMON Modify Defaults option

PROMON utility

Starts the monitor that displays database information.

Syntax

```
promon db-name
```

Parameters

db-name

Specifies the database you want to monitor.

When you enter the PROMON utility, the main menu appears, as shown in the following figure.

```
                    OpenEdge MONITOR Release 12

                    Database: /usr/wrk/sports2000

                    1. User Control
                    2. Locking and Waiting Statistics
                    3. Block Access
                    4. Record Locking Table
                    5. Activity
                    6. Shared Resources
                    7. Database Status
                    8. Shut Down Database
                    9. Currently Connected Tenants

                  R&D. Advanced Options
                    T. 2PC Transactions Control
                    L. Resolve 2PC Limbo Transactions
                    C. 2PC Coordinator Information

                    J. Resolve JTA Transactions

                    M. Modify Defaults
                    Q. Quit

                    Enter your selection:
```

PROMON utility main menu

The sections that follow describe each PROMON option.

If you choose **options 1** to **4** or **option T**, the following menu appears:

```
                    1. Display all entries
                    2. Match a user number
                    3. Match a range of user numbers
                    4. Match a record id
                    5. Change list sorting
                    6. Match a tenant or tenants
                    7. Match a partition or partitions
                    Q. Return to main menu

                    Enter your selection:
```

Note: The entire selection list does not always appear. Only the choices that are relevant to your first selection appear.

After the first screen of output from any option, press **U** to monitor the same parameters continuously. The default interval is eight seconds, but you can change it on the **Modify Defaults** screen (**option M** on the main menu).

Notes

- To collect output data, the monitor bars all users from shared memory for an instant and takes a snapshot of its contents. Therefore, do not use the monitor when system performance is being measured.
- Certain monitor output data is not generally useful, but it might be helpful when debugging unusual system problems.

Viewing the AUS Server status

To view the AUS Server status, enter `R&D`. The following menu appears.

Note: For more information, refer Monitor AUS Server activities in *OpenEdge ® Data Management: SQL Development* guide.

```
 11/05/18        OpenEdge Release 12 Monitor (R&D)
 14:31:39        Main (Top) Menu

                  1. Status Displays ...
                  2. Activity Displays ...
                  3. Other Displays ...
                  4. Administrative Functions ...
                  5. Adjust Monitor Options

 Enter a number, <return>, P, T, or X (? for help):
```

Enter `4` to navigate to the **Administrative Functions** menu.

PROMON User Control option

Displays information for each database process.

Sample output for PROMON User Control option

The following shows an example of this option's output. Not all fields in the User Control display fit in a standard window width, but all are described below.

```
User Control: by user number

  Usr:Ten  Name      Domain     Type     Wait  Table:Part  Dbkey  Trans    PID  Sem Srv   Login
  Time IVP#  Remote Address
    0:0    dba                  BROK     --       0:0          0      0   6492   0    0
06/24/14
    5:3    lobtenant lobtenant  SELF/ABL --       0:0          0      0   7364   0    0
06/24/14
    6:4    lobtenant lobtenant  SELF/ABL --       0:0          0      0   4780   0    0
06/24/14
    7:2    tenant2   tenant2    SELF/ABL --       0:0          0      0   6932   0    0
06/24/14
    8:1    tenant1   tenant1    SELF/ABL --       0:0          0      0   3616   0    0
06/24/14
    9:0    dba                  MON      --       0:0          0      0   7892   0    0
06/24/14
   23:0                   0     REMC/ABL --      31:47         0  98676      0  11   2
06/24/14 01:05 IPV4 172.16.18.92

RETURN - repeat, U - continue uninterrupted, Q - quit
```

`Usr:Ten`

Displays the user number and tenant ID.

User numbers are assigned sequentially to database processes as they are started. The broker process is always user (0).

`Name`

The user name for end-user (client) processes.

`Domain`

The user's domain name.

`Type`

The table below lists the distinct types of database processes that can appear in the Type field.

Table 80: Process type values

Value	Process type
AIMD	After-image file management utility daemon
AIW	After-image writer (AIW) process
APW	Asynchronous page writer
AUDA	Audit archive utility
AUDL	Audit load utility

Value	Process type
BAT	Batch user
BIND	Binary dump utility
BINL	Binary load utility
BIW	Before-image writer
BKUP	Online backup
BROK	Broker process
FMA	OpenEdge Management (formerly Fathom™ Management) Monitor
IACT	Index activate utility
IDXC	Index compact utility
IDXF	Index fix utility
IDXM	Index move utility
MON	Monitor process
PASA	Progress Application Server for OpenEdge Session Manager
PASN	Progress Application Server for OpenEdge Session
PSA	PROSTRCT ADD utility
QUIE	PROQUIET utility
REMC	Remote client
RFUT	RFUTIL utility
RPLA	OpenEdge Replication Site Replication Agent
RPLE	OpenEdge Replication Site Replication online enable
RPLS	OpenEdge Replication Site Replication Server
RPLT	OpenEdge Replication Site Replication Transition
RPLU	OpenEdge Replication Site Replication Utilities
SELF	Self-service client
SERV	Remote-user server
SHUT	Database shutdown process

Value	Process type
SQSC	SQLSCHEMA utility
SQDP	SQLDUMP utility
SQLD	SQLLOAD utility
SQSV	SQL server
TBLM	Table move utility
WDOG	Watchdog

`Wait`

Indicates that a process is waiting for a lock or latch. The table below lists common values and describes the resource or event associated with each value.

Table 81: Possible wait column values

Value	Description
Lock waits	
REC	Row (record) lock
IX	Index lock
SCHE	Schema lock
TRAN	Transaction commit
TXB	Transaction end update mode lock
TXE	Transaction end commit mode lock
TXS	Transaction end share lock
TXX	Transaction end exclusive lock
Resource waits	
AIRD	After-image buffer read from disk
AIWR	After-image buffer write to disk
BIRD	Before-image buffer read from disk
BIWR	Before-image buffer write to disk
BKEX	Exclusive lock on database buffer

Value	Description
BKSH	Share lock on database buffer
BKSI	Share lock with intent to update on database buffer
BUFF	Shared buffer
DB	Database server
DBBK	Database buffer being backed up
DBRD	Database buffer read from disk
DBWR	Database buffer write to disk
DEAD	Process is waiting to be removed from the process list[33]
RGET	Row get lock
SRPL	Replication agent acknowledgment
Latch waits	
AIB	After-image buffer latch
BFP	Buffer pool latch
BHT	Buffer pool hash table latch
BUF	Buffer pool buffer latch
CPQ	Checkpoint queue latch
DLC	Index delete chain
GST	Global storage pool latch
IXD	Index delete chain latch
LKF	Lock table free chain latch
LKP	Lock table purge chain latch
LKT	Lock table latch
LRS	Secondary LRU chain (not currently implemented)
LRU	Buffer pool LRU chain latch
MTL_RPL	Replication latch

[33] Seen when a process is terminated with the UNIX `kill -9` command

Value	Description
MTX	Log space allocation latch
PWQ	Page writer queue latch
SCH	Shared schema cache latch
SEQ	Sequence cache latch
TXQ	Transaction queue latch
TXT	Transaction table latch
USR	User table latch

You may also see the following values listed in the **Wait** column:

- BF2
- BF3
- BF4
- BIB
- LRI
- LRX
- OM

The previous values are for short-duration (micro- to milliseconds) latches used internally by the OpenEdge RDBMS.

`Table:Part Dbkey`

If the user has a REC wait, the combination of the table number, partition number, and the db key, identifies the record. If the table is not table partitioned, the partition number is 0.

`Trans`

Transaction (task) number, if one is active. After the broker starts, numbers are assigned sequentially to each transaction that accesses the database.

`PID`

The process ID as assigned by the operating system. The PID column typically displays (0) for remote clients.

`Sem`

The number of the semaphore the process is using. Each process uses exactly one semaphore. The database engine uses two semaphores (numbers 1 and 2) internally and assigns the remaining semaphores one at a time (starting with the broker) as database processes log in.

`Srv`

For remote clients (**REMC**), the user number of the server the client runs against.

Login

The date the process started.

Time

The time the process started.

IPV#

The IP version.

Remote Address

PROMON Locking and Waiting statistics option

Displays lock and wait information for each database process. The first two lines display cumulative statistics (since the broker started) for all users.

Sample output for PROMON Locking and Waiting statistics option

The following shows an example of this option's output.

```
Locking and Waiting:
Type   Usr:Ten   Name     Domain     Record     Trans     Schema
Lock 99999       TOTAL...               864         0          0
Wait 99999       TOTAL...                11         0          0
Lock     0:0     dba                      0         0          0
Wait     0:0     dba                      0         0          0
Lock     5:1     tenant1   tenant1       12         0          0
Wait     5:1     tenant1   tenant1        0         0          0
Lock     6:3     lobtenant lobtenant     12         0          0
Wait     6:3     lobtenant lobtenant      0         0          0
Lock     7:2     tenant2   tenant2       15         0          0
Wait     7:2     tenant2   tenant2        0         0          0
Lock     8:4     lobtenant lobtenant     13         0          0
Wait     8:4     lobtenant lobtenant      0         0          0
Lock     9:0     dba                      0         0          0
Wait     9:0     dba                      0         0          0

RETURN - show remaining, U - continue uninterrupted, Q - quit:
```

Type

Lock or Wait. The output display includes two lines for each process. The first shows the number of record, transaction, and schema locks obtained, and the second shows the number of times the process waited to obtain one of these locks.

Usr:Ten

The user number and tenant ID of the process. User number 999 represents all users.

Name

For client processes, the user name.

Domain

The user's domain name.

Record

For the Lock type, the total number of record locks obtained. For the Wait type, the number of times the user had to wait to obtain a record lock. Numbers are cumulative for the life of the process.

Trans

For the Lock type, the number of times a transaction lock was issued. A transaction lock is issued for the duration of a transaction when a record is deleted. If the transaction is ultimately undone or backed out, the record is not deleted.

For the Wait type, the number of times the user had to wait before accessing or skipping a record marked for deletion by an unfinished transaction. This transaction wait corresponds exactly to the TSK wait described under User Control (option 1).

Schema

For the Lock type, the total number of times the schema lock was obtained. For the Wait type, the number of times the user had to wait to obtain the schema lock. (There is one schema lock; only one user at a time can update any part of the schema.)

PROMON Block Access option

Displays database buffer pool access statistics. The following shows an example of this option's output.

Sample output for PROMON Block Access option

```
Block Access:
Type   Usr:Ten  Name      Domain    DB Requests       DB Reads       BI Reads      AI
Reads
                                                    \Writes        \Writes       \Writes
Acc  99999      TOTAL...                 13604           72             47            0
                                                          4              3            0
Acc      0:0    dba                        682           34             47            0
                                                          3              2            0
Acc      5:1    tenant1    tenant1         607            0              0            0
                                                          0              0            0
Acc      6:3    lobtenant  lobtenant       607            0              0            0
                                                          0              0            0
Acc      7:4    lobtenant  lobtenant       647            0              0            0
                                                          0              0            0
Acc      8:2    tenant2    tenant2        647            0              0            0
                                                          0              0            0
RETURN - repeat, U - continue uninterrupted, Q - quit:
```

The first line displays cumulative information for all users. The six read and write columns refer to disk I/O. Reads and writes are always one block. Block size varies among systems, but is usually 512 bytes, 1,024 bytes, or 2,048 bytes.

Type

This column always has the value Acc.

`Usr:Ten`

The user number and tenant ID of the process.

`Name`

For client processes, the user name.

`Domain`

The user's domain name.

`DB Request`

The number of times the database buffer system was searched to find a block. The buffer system is searched every time a process attempts to access a record. If the block that holds the desired record is already in memory, a disk read is not required. If the ratio of DB Reqst to DB Read is not high (10 to 1 or greater), consider raising the value of the Blocks in Database Buffers (`-B`) startup parameter. Regardless of the number of available buffers, random record access causes a lower database request to disk read ratio than sequential record access.

`DB Read`

The number of database disk block reads. A database block must be read from disk when a process accesses a record whose containing block is not already in the database buffers. Recall that for read-only database requests, the OpenEdge RDBMS uses private database buffers if they are available, rather than the shared buffer pool (allocated with the Blocks in Database Buffers (`-B`) parameter).

`DB Write`

The number of database block writes to disk. Once the database buffers are full, every disk read overwrites an existing block; if the overwritten block has been modified since it was read in, it must be written to disk. This accounts for the majority of block writes.

`BI Read`

The number of before-image (BI) file block reads. For example, the BI file is read when a transaction is undone. The BI file has its own one-block input buffer and does not use the database buffer pool.

`BI Write`

The number of BI file block writes. When a record is updated, a pretransaction copy of the record is written to the BI file. When the transaction completes, the database engine writes the last BI file block out to disk (assuming you are running the database with full integrity). This post-transaction disk write accounts for the relatively high number of BI file writes, but it can be delayed with the Delay BI File Write (`-Mf`) startup parameter.

In addition to record images, the database engine writes to the BI file various notes and data required to reconstruct a damaged database. The BI file has its own one-block output buffer and does not use the shared database buffer pool.

`AI Read`

The number of after-image (AI) file block reads. The AI file is read during crash recovery. The AI file has a one-block input/output buffer and does not use the database buffer pool.

AI Write

The number of AI file block writes. When you run the database with after-imaging enabled, a copy of each note written to the BI file is written to the AI file.

PROMON Record Locking Table option

Displays the contents of the record locking table. Each line corresponds to an entry in the locking table. The lock table contains entries for all record locks currently granted or requested by users of the database.

Sample output for PROMON Record Locking Table option

The following shows an example of this option's output.

```
Record Locking Table:by user number
Usr:Ten  Name      Domain    Chain #    Row-id   Table:Part  Lock Flags  Tran State Tran
 ID
  5:2    tenant2   tenant2   REC  517    4229       -1:0     EXCL        None          0
  8:4    lobtenant lobtenant REC  655    4367       -1:0     EXCL        None          0
RETURN - repeat, U - continue uninterrupted, Q - quit
```

The size of the record locking table is set with the Locking Table Entries (-L) startup parameter. See the chapter on locks in *Develop ABL Applications* for more information on locks and locking.

Usr:Ten

The user number and tenant ID.

Name

For client processes, the user name; for a lock from a JTA transaction, <JTA> is displayed.

Domain

The user's domain name.

Chain

The chain type should always be REC, the record lock chain.

#

The record lock chain number. The locking table is divided into chains anchored in a hash table. These chains provide for fast lookup of record locks by Rec-id.

Row-id

The row-id for the lock table entry.

Table:Part

The ID and Partition ID of the locked table.

`Lock`

One of five lock types: X (exclusive lock), S (share lock), IX (intent exclusive lock), IS (intent share lock), or SIX (shared lock on table with intent to set exclusive locks on records).

`Flags`

There are five possible types of flags. The table below lists the flags and their meanings.

Table 82: Flag values

Flag	Name	Description
C	Create	The lock is in create mode.
D	Downgrade	The lock is downgraded.
E	Expired	The lock wait timeout has expired on this queued lock.
H	No hold	The "nohold" flag is set.
J	JTA	The lock is part of a JTA transaction
K	Keep	Keep the lock across transaction end boundary
L	Limbo lock	The client has released the record, but the transaction has not completed. The record lock is not released until the transaction ends.
P	Purged lock entry	The lock is no longer held.
Q	Queued lock request	Represents a queued request for a lock already held by another process.
U	Upgrade request	The user has requested a lock upgrade from SHARE to EXCLUSIVE.

`Trans State`

The state of the transaction. The table below lists the possible states.

Table 83: Transaction states

State	Description
Begin	A transaction table entry was allocated, and a start record is being logged
Active	The transaction is doing forward processing
None	The lock was acquired outside of a transaction
Dead	The transaction is complete, but the lock has not been released
Prep	The transaction is preparing to enter phase 1 (ready to commit), but has not sent a ready-to-commit reply

State	Description
Phase 1	In phase 1, ready to commit
Phase 2	In phase 2
C	With two-phase commit, this user is logging for the coordinator for the transaction
R	Ready to commit
L	Limbo transaction
Active JTA	The transaction is currently executing
Idle JTA	The transaction is not currently executing
Prepared JTA	The transaction is prepared
RollbackOnly JTA	The transaction has encountered an error
Committed JTA	The transaction is in the commit process

`Trans ID`

Transaction ID of the lock.

PROMON Activity option

Displays an overview of your system activity.

Sample Output for PROMON Activity option

The following shows an example of this option's output.

```
Activity  - Sampled at 04/29/04 15:09 for 0:03:39.

Event                Total   Per Sec   Event              Total     Per Sec
       Commits          50       0.2              Undos       0         0.0
Record Updates           0       0.0       Record Reads    3709        16.9
Record Creates           0       0.0     Record Deletes       0         0.0
     DB Writes           3       0.0           DB Read      473         2.2
     BI Writes           2       0.0           BI Reads     198         0.9
     AI Writes           0       0.0
  Record Locks         210       1.0       Record Waits       0         0.0
   Checkpoints           0       0.0      Buffs Flushed       0         0.0

Rec Lock Waits    0 %    BI Buf Waits        0 %    AI Buf Waits    0 %
Writes by APW     0 %    Writes by BIW       0 %    Writes by AIW   0 %
Buffer Hits      99 %    Primary Hits      100 %    Alternate Hits 99 %
DB Size        1151 K        BI Size    2168 K        AI Size       0 K
FR chain         12 blocks   RM chain    439 blocks
Shared Memory 966656         Segments      1

0 Servers, 10 Users (10 Local, 0 Remote, 10 Batch),0 Apws

RETURN - repeat, U - continue uninterrupted, Q - quit:
```

`Event`

The events that have occurred on the system and the cumulative total number of events and the number of events per second. The table below defines the event types listed in the **Event** field. For each event type, PROMON lists the cumulative total number of events and the number of events per second.

Table 84: Event types

Event type	Description
Commits	Number of transactions committed
Undos	Number of transactions rolled back
Record Updates	Number of records updated
Record Reads	Number of records read
Record Creates	Number of records created
Record Deletes	Number of records deleted
DB Writes	Number of database blocks written to disk
DB Reads	Number of database blocks read from disk
BI Writes	Number of BI blocks written to disk
BI Reads	Number of BI blocks read

Event type	Description
AI Writes	Number of AI blocks written to disk
Record Locks	Number of record locks used
Record Waits	Number of times the database engine waited to access a locked record
Checkpoints	Number of checkpoints performed
Buffers Flushed	Number of database buffers flushed during checkpoints

`Rec Lock Waits`

Percentage of record accesses that result in record lock waits. A record lock wait occurs when the database engine must wait to access a locked record.

`BI Buf Waits`

Percentage of before-image (BI) buffer waits. A BI buffer wait occurs when the database engine must wait to access a BI buffer.

`AI Buf Waits`

Percentage of after-image (AI) buffer waits. An AI buffer wait occurs when the database engine must wait to access an AI buffer.

`Writes by APW`

Percentage of database blocks written to disk by the APW; this is a percentage of the total number of database blocks written by the database engine.

`Writes by BIW`

Percentage of BI blocks written to disk by the BIW; this is a percentage of the total number of BI blocks written to disk by the database engine.

`Writes by AIW`

Percentage of AI blocks written to disk by the AIW; this is a percentage of the total number of AI blocks written.

`Buffer Hits`

Percentage of buffer hits for both the primary and Alternate Buffer Pools. A buffer hit occurs when the database engine locates a record in the buffer pool and does not have to read the record from disk.

`Primary Hits`

Percentage of buffer hits for the primary buffer pool.

`Alternate Hits`

Percentage of buffer hits for the Alternate Buffer Pool.

`DB Size`

Size of your database in kilobytes.

`BI Size`

Size of your BI file in kilobytes.

`AI Size`

Size of your AI file in kilobytes.

`FR chain`

Number of blocks on your database's free chain. The free chain is a chain of empty database blocks.

`RM chain`

Number of blocks on your database's record-management chain. The record-management chain is a chain of partially filled database blocks.

`Shared Memory`

Amount of shared memory used by the database, in kilobytes.

`Segments`

Number of shared-memory segments allocated by the database engine. The size of a shared-memory segment is determined by your kernel configuration.

The last line of the PROMON Activity output summarizes the current number of each type of process running against the database at the time you ran the Activity option, not a cumulative total. The table below defines the process types.

Table 85: Process types

Field	Description
Servers	Number of servers running against your database. This is the current value at the time you ran the Activity option, not a cumulative total.
Users	Number of users running an OpenEdge process. This field is further divided into the type of user: Local, Remote, or Batch. This is the current value at the time you ran the Activity option, not a cumulative total.
Apws	Number of APWs running against your database. This is the current value at the time you ran the Activity option, not a cumulative total.

PROMON Shared Resources option

Displays a variety of system resource usage statistics and all startup parameter settings. Broker status information appears only when a broker process is active. See Database Startup Parameters on page 473 for more information on these startup parameters.

Sample Output for PROMON Shared Resources option

The following shows this option's output.

```
Shared Resources:
                                          Shared memory version number: 6413723
                                                 Busy After Image Extent:
/scratch/doc/117/testdb.a3
                                         Locking table entries in use: 0
                                        Locking table high water mark: 1
                                            Number of semaphores used: 33
                                                        Broker status: Executing
                                                   BI Writer status: Not executing
                                                   AI Writer status: Not executing
                                                    Watchdog status: Not executing
                                              Number of page writers: 0
                                         Number of self-service users: 0
                                              Number of remote users: 0
                                                   Number of servers: 0
                                                 Number of shut-downs: 0
                                                   Number of monitors: 1
            After-image Management Archival Directory List (-aiarcdir): Not Enabled
   Create After-image Management Archival Directory(s) (-aiarcdircreate): Not Enabled
              After-image Management Archival Interval (-aiarcinterval): -1
                                 Number of After-Image Buffers (-aibufs): 20
                                          After-Image Stall (-aistall): Not Enabled
                Starting index number for statistics range (-baseindex): 1
                Starting table number for statistics range (-basetable): 1
    Starting index number per user for statistics range (-baseuserindex): 1
    Starting table number per user for statistics range (-baseusertable): 1
                               Number of Before-Image Buffers (-bibufs): 20
                                     BI File Threshold Stall (-bistall): Disabled.
                                      BI File Threshold size (-bithold): 0.0   Bytes
                                        Database Blocksize (-blocksize): 4096
                                            BIW writer delay (-bwdelay): 0
                                             Allowed index cursors (-c): 84
                                              CDC cache size (-cdcsize): 200k bytes
                           SSL Certificate Store Path (-certstorepath): Not Enabled
                                           Character Set (-cpinternal): iso8859-1
                                         Physical Database Name (-db): jan28
                                       Diagnostic directory (-diagDir): Not Enabled
      Diagnostic events value (-diagEvent): LockTable:0,BiThold:0,SysErr:0
                                 Diagnostic event level (-diagEvtLevel): 0
                                   Diagnostic field separator (-diagFS): ' '
                                   Diagnostic data format (-diagFormat): csv
                                   Diagnostic pause length (-diagPause): 0
                                 Diagnostic prefix value (-diagPrefix): diagEvent_
                                      Diagnostic procedure (-diagProc): Not Enabled
                                                Direct I/O (-directio): Not Enabled
                                                   Database Type (-dt): PROGRESS
                                     Encryption cache size (-ecsize): 1000
                                           Group delay (-groupdelay): 10
                                         Hash Table Entries (-hash): 887
                                                 Crash Recovery (-i): Enabled
   Number of indexes included in statistics collection (-indexrangesize): 50
                                              TCP/IP Version (-ipver): Not Enabled
                                       SSL Key Alias Name (-keyalias): Not Enabled
                                  Lock table hash table size (-lkhash): 1237
                               Original Lock Release Algorithm (-lkrela): Not Enabled
                                   Number of LRU force skips (-lruskips): 0
                                 Number of LRU2 force skips (-lru2skips): 0
                                      Maximum Area Number (-maxArea): 32000
                                Size of JTA transaction table (-maxxids):  100
                                Maximum Port for Auto Servers (-maxport): 0
                                Minimum Port for Auto Servers (-minport): 0
                          Multi-tenancy partition cache size (-mtpmsize): 1024
                                           Use muxlatches (-mux): 1
                                       Maximum Number of Users (-n): 21
```

```
                    Minimum time to nap at first -spin exhaustion (-nap): 10
                       Maximum time to nap at -spin exhaustion (-napmax): 250
                                    SSL No Host Verify (-nohostverify): Not Enabled
                                        Disable LRU mechanism (-nolru): Not Enabled
                                  SSL No Session Cache (-nosessioncache): Not Enabled
                                  SSL No Session Reuse (-nosessionreuse): Not Enabled
                                       Login Governor (-nGovernor): 0 of 21
          Number of checkpoint statistics to record (-numCheckpointStats): 32
                                    Storage object cache size (-omsize): 1024
                       Database Service Manager - IPC Queue Size (-pica): 64.0  KBytes
                                 Shared memory segments locked (-pinshm): Not Enabled
                      Use pollset mechanism for client/server (-pollset): Not Enabled
                           Delay first prefetch message (-prefetchDelay): Not Enabled
                      Prefetch message fill percentage (-prefetchFactor): 0
                      Minimum records in prefetch msg (-prefetchNumRecs): 16
                      Suspension queue poll priority (-prefetchPriority): 0
                  APW queue scan cycle time in milliseconds (-pwqdelay): 100
                        APW minimum queue length before write (-pwqmin): 1
                     APW buffer scan cycle time in seconds (-pwsdelay): 1
               APW maximum number of buffers to scan per cycle (-pwscan): 5
             APW maximum number of buffers to write per cycle (-pwwmax): 25
                                      Before-Image File I/O (-r -R): Reliable
        Record free chain search depth factor (-recspacesearchdepth): 5
                                    Security cache size (-secsize): 512
                                 Number of Semaphore Sets (-semsets): 3
                               SSL Session Timeout (-sessiontimeout): 0
                    Maximum Shared Memory Segment Size (-shmsegsize): 0 Mb
                                  Current Spin Lock Tries (-spin): 24000
                         SSL Encryption for TCP/IP connections (-ssl): Not Enabled
                    STS Debug Logging Level default (-stsloggginglevel): 0
  Number of tables included in statistics collection (-tablerangesize): 50
                                User Notification Time (-usernotifytime): 0 seconds
Number of indexes per user included in statistics collection (-userindexrangesize): 50
Number of tables per user included in statistics collection (-usertablerangesize): 50
                             Area block consistency check (-AreaCheck): Not Enabled
                                  Number of Database Buffers (-B): 3000
                           Number of Alternate Database Buffers (-B2): 0
                            Maximum private buffers per user (-Bpmax): 64
                          Database block consistency check (-DbCheck): Not Enabled
          Database Service Manager - Service(s) to start (-DBService): Not Enabled
                                    Enhanced Read-Only mode (-ERO): Not Enabled
                                           Force Access (-F): Not Enabled
                                Before-Image Truncate Interval (-G): 0
                                             Host Name (-H): Not Enabled
                         Index block consistency check (-IndexCheck): Not Enabled
                                  Current Size of Lock Table (-L): 8192
                                       Lock Governor (-LGovernor): 0%
                          Maximum Number of Clients Per Server (-Ma): 4
                                  Memory overwrite check (-MemCheck): Not Enabled
                                   Delay of Before-Image Flush (-Mf): 3
                                    Minimum Clients Per Server (-Mi): 0
                                          Message Buffer Size (-Mm): 0
                                   Maximum Number of Servers (-Mn): 5
                                        Servers per Protocol (-Mp): 0
                                  Maximum Servers Per Broker (-Mpb): 0
                                   Excess Shared Memory Size (-Mxs): 26
                                           Network Type (-N):
                        Server network message wait time (-Nmsgwait): 2
                  Pending client connection timeout (-PendConnTimeout): 0
                                           Service Name (-S): 0
                             Broker server group support (-ServerType): BOTH
                             SQL Server Max Open Cursors (-SQLCursors): 0
                                  SQL Server Stack Size (-SQLStack): 0
                      SQL Server Statement Cache Size (-SQLStmtCache): 0
Size [1K byte units] of SQL Server temp table buffer (-SQLTempStoreBuff): 0
Size [1K byte units] of SQL Server temp table disk storage (-SQLTempStoreDisk): 0
Size [1K byte units] of SQL Server temp table data page (-SQLTempStorePageSize): 0
                     Authorized data truncation (-SQLTruncateTooLarge): OFF
                        SQL Autonomous Schema Update (-SQLWidthUpdate): OFF
```

```
                          Record block consistency check (-TableCheck): Not Enabled
                                TXE Lock retry limit (-TXERetryLimit): 0
                             TXE Commit lock skip limit (-TXESkipLimit): 10000

RETURN - repeat, U - continue uninterrupted, Q - quit:
```

Because most of the Shared Resources options output is self-explanatory, this list describes only two items:

`Shared memory version number`

The version number of the shared-memory data structure. This structure varies slightly between some releases of the database. Since broker, server, and client processes all access the same shared-memory pool, their shared-memory version numbers must agree. If an error message states that shared-memory version numbers do not match, make sure that the broker and clients are running the same OpenEdge release.

`Number of semaphores used`

Shows the number of semaphores preallocated for this database. Because each process requires one semaphore, semaphores are preallocated based on the number of processes expected to access the database.

For a particular database, the number of semaphores used (#SEM) is as follows:

```
#SEM = Max-possible-users (-n) + Max-possible-servers (-Mn) + 4
```

The maximum number of semaphores you can use on your system is set with a kernel configuration parameter (`SEMMNS`). If the number of database users grows large, you can exceed this value. You must reconfigure the system's kernel to increase the semaphore limit. (See Managing Performance on page 275 for more information.) You can only reduce semaphore use by lowering the values of the Number of Users (`-n`) or Maximum Servers (`-Mn`) startup parameters. (When you start the broker process, a message specifies the number of semaphores still available.)

PROMON Database Status option

Displays database block usage statistics and general status information.

Sample output for PROMON Database Status option

The following shows an example of this option's output.

```
                    Database version number: 4266
                             Database state: Open (1)
                      Database damaged flags: None (0)
                             Integrity flags: Executing with -r (0)
                 Database block size (bytes): 4096
               Total number of database blocks: 1417
               Database blocks high water mark: 1375
            Free blocks below high water mark: 116
                Record blocks with free space: 3
              Before image block size (bytes): 8192
               Before-image cluster size (kb): 512
               After image block size (bytes): 8192
                         Last transaction number: 169
```

```
                   Highest file number defined: 0
                Database create (multi-volume): 08/04/11 16:05
                     Most recent database open: 08/04/11 14:19
                        Previous database open: 08/04/11 14:19
                 Local Schema cache time stamp: 08/04/11 20:47
                     Most recent .bi file open: 08/04/11 14:20
                        Previous .bi file open: - -
                   Time since last truncate bi: 0 (seconds)
                       Most recent full backup: 08/03/11 15:06
                 Database changed since backup: Yes
               Most recent incremental backup: - -
                  Sequence of last incremental: 0
                        Database Restrict Mode: Disabled

      RETURN - repeat, U - continue uninterrupted, Q - quit
```

Because most of the Database Status option output is self-explanatory, this list describes only five items:

`Database state`

The state of the database, as set in the database master block:

Database state	Description
1	The database is open
2	The database is not modified since the last open
4	The database is being recovered
8	A database restore is underway
2048	An index repair is underway

`Total number of database blocks`

The actual database size in disk blocks.

`Number of empty blocks`

The number of empty blocks in the database. Empty blocks are created when the database outgrows its existing block allocation, and also by record and index deletion. The database engine expands the database by multiple blocks at a time—usually 8 or 16—and most of these are empty at first. Typically, the number of empty blocks increases and decreases as the database grows.

`Record blocks with free space`

Free space is created when records are deleted and when the database grows into a new block, but does not use the entire block. The database engine reuses empty space with new records when possible, but does not return empty space to the operating system. This space does not affect database performance. However, if you must reclaim this disk space for your system, you can dump and reload the database.

`Time since last truncate bi`

The time in seconds since the before-image file was truncated. When the BI file is truncated, all modified database blocks are written to disk.

PROMON Shut Down Database option

Lets you disconnect a user or shut down a database.

Sample output for PROMON Shut Down Database option

When you choose this option, PROMON displays the menu shown below.

```
Enter your selection:

            1 Disconnect a User
            2 Unconditional Shutdown
            3 Emergency Shutdown (Kill All)
            x Exit

Enter choice>
```

`Disconnect a User`

Disconnects a single user from the database specified when PROMON was invoked. PROMON prompts you to enter the user number of the user you want to disconnect. (PROMON displays the user number of active users above this submenu.) Enter the user number, then press **RETURN**.

`Unconditional Shutdown`

Shuts down the database in an orderly fashion. When you choose `Unconditional Shutdown`, PROMON begins the shutdown when you press **RETURN**. PROMON does not prompt you to confirm the shutdown.

`Emergency Shutdown (Kill All)`

Shuts down the database without performing cleanup and kills all processes connected to the database. Before performing the shutdown, PROMON displays the following message prompting you to confirm the shutdown:

```
This is rather extreme. All the processes connected to
the database will be destroyed. Do you really want to?
Please confirm by entering y for yes:
```

If you enter anything other than **y** or **yes**, PROMON cancels the shutdown and returns to the PROMON utility main menu.

PROMON Currently Connected Tenants option

Displays information for each currently connected tenant.

Sample output from PROMON Currently Connected Tenants option

The following shows an example of this option's output.

```
08/04/11        Currently Connected Tenants
17:59:35

 Sel  Tenant Id      Name                                  User Count
               0      Default                                     2
               1      tenant1                                     1
               2      tenant2                                     1
               3      lobTenant1                                  1
               4      lobTenant2                                  1

Enter <return>, R, U, P, T, or Q (? for help):
```

`Sel`

Indicates that the tenant is selected in the filtering process. See R&D Adjust Monitor Options on page 622 for information on filtering tenants.

PROMON R&D Advanced Options

Provides advanced options for in-depth monitoring of OpenEdge RDBMS activity and performance. These options are called R&D options.

To access PROMON R&D options, at the **Enter your selection** prompt, enter **R&D**. The R&D main menu appears:

```
09/19/08          OpenEdge Release 11 Monitor (R&D)
14:50             Main (Top) Menu
         1. Status Displays ...
         2. Activity Displays ...
         3. Other Displays ...
         4. Administrative Functions ...
         5. Adjust Monitor Options

Enter a number, <return>, P, T, or X (? for help):
```

Choose one of the following options:

- **Status Displays** — Shows information about the current state of the database and its users.
- **Activity Displays** — Shows information about database activity in the recent past.

  Activity displays shows the total number of each type of operation for the sample period, the number of operations per minute, the number of operations per second, and the number of operations per transaction.
- **Other Displays** — Shows information that does not fit into either the **Status** or **Activity** categories.
- **Administrative Functions** — Allows you see and change options related to administering a database.
- **Adjust Monitor Options** — Allows you change the way the monitor behaves.

Note:

Menus items followed by ellipses (...) access a lower-level menu.

R&D Status Displays

The **R&D Status** displays provide tables of information regarding the current state of the database.

Status can be queried for the following:

- Database on page 566
- Backup on page 568
- Servers on page 569
- Processes/Clients on page 570
- Files on page 574
- Lock Table on page 575
- Buffer Cache on page 576
- Logging Summary on page 577
- BI Log on page 577
- AI Log on page 578
- Two-phase Commit on page 579
- Startup Parameters on page 580
- Shared Resources on page 582
- Shared Memory Segments on page 583
- AI Extents on page 583
- Database Service Manager on page 585
- Servers by broker on page 586
- Client Database-Request Statement Cache on page 587
- Schema Locks & Wait Queue on page 592
- Broker Startup Parameters on page 592

Database

Displays general database status information.

PROMON Database Status display

The following shows a sample Database Status display.

```
09/15/16            Status: database
11:01:01

Database was started at:          09/15/16 9:17
It has been up for:               1:43:37
Database state:                   Open (1)
Database damaged flags:           None
Integrity flags:                  None
Most recent database open:        09/15/16 9:17
Previous database open:           09/15/16 9:17
Local cache file time stamp:      09/10/16 21:41
Database block size:              4029 bytes
Number of blocks allocated:       324 (1268 kb)
Empty blocks:                     20 (6%)
Free blocks:                      0 (0%)
RM Blocks with Free Space:        4 (1%)
Last transaction id:              172
Highest table number defined:     9
Database version number:          143
Shared memory version number:     13018
Database Restrict Mode:           Disabled
```

The display shown above contains the following fields:

- **Database state** — The current operating mode of the database. The table below describes the possible database states.

  Table 86: Database states

State	Description
Open	The database is open
Not modified since last open	The database is closed
Recovery	The database is performing crash recovery
Index repair	The PROUTIL IDXBUILD utility is running to rebuild indexes
Restore	The database is being restored

- **Database damaged flags** — The state of the database. The table below lists the possible flags.

  Table 87: Database damaged flags

State	Description
None	Normal
Opened with -F	Crash recovery skipped; the database might be damaged
Crashed with -i	Crashed without integrity; the database is damaged
Crashed with -r	Crashed with buffered I/O; the database might be damaged

- **Integrity flags** — The integrity mode in which the database is running. The table below lists the possible flags.

  Table 88: Integrity flags

State	Description
None	Normal
Executing with -i	A crash in this state will damage the database
Executing with -r	A crash in this state might damage the database

- **Most recent database open** — The date and time when the broker for this database started.
- **Previous database open** — The date and time when the database previously started.
- **Local cache file time stamp** — The time stamp used to check the validity of the schema cache file. See the description of the Schema Cache File (`-cache`) parameter in Database Startup Parameters on page 473 for more information.
- **Database block size** — The database block size in bytes.
- **Number of blocks allocated** — The total number of blocks allocated to the database.
- **RM Blocks with free space** — The total number of blocks in the RM chain.
- **Highest table number defined** — The number of tables defined in the database.
- **Database version number** — Identifies the structures of all database-related data that is resident on disk. This number is used to match different executable versions with the correct database structures.
- **Shared memory version number** — Identifies the version of all data structures that are resident in shared memory and used by the database manager. This number is used to ensure that all executables accessing shared memory agree on what data is kept in shared memory and how that data is accessed. The shared-memory version number changes more frequently than the database version number.
- **Database Restrict Mode** — Identifies if the database is in a restricted access mode.

Backup

Displays information about database backups.

PROMON Backup Status display

The following shows a sample Backup Status display.

```
10/18/06            Status: backup
16:07

Most recent full backup:            10/04/06 11:52
Most recent incremental backup:     Never
Database changed since backup:      Yes
Sequence of last incremental:       0
```

The display shown above contains the following fields:

- **Most recent full backup** — The date of the most recent full backup

- **Most recent incremental backup** — The date of the most recent incremental backup
- **Database changed since backup** — Yes, if the database has been modified since the last backup. No, if the database has not changed
- **Sequence of last incremental** — The count of incremental backups made since the last full backup

Servers

Displays status information about OpenEdge servers running on the system.

PROMON Servers Status display

The following shows a sample Servers Status display.

```
09/15/05        Status: Servers
14:07:47

Sv                                      Pend.   Cur.   Max.   Port
No    Pid  Type       Protocol Logins  Users  Users  Users    Num

  0   4899 Login      TCP           1      0      0      5   2402
  1  22849 Auto       TCP           1      0      1      5   1025
  2      0 Inactive                 0      0      0      0      0
  3      0 Inactive                 0      0      0      0      0
  4      0 Inactive                 0      0      0      0      0
```

The display contains the following fields:

- **Type** — The server type. The table below describes the possible server types.

  Table 89: Server types

State	Description
Broker	A broker process. A broker logs in clients and starts remote servers.
Auto	A server started automatically by the broker.
Login	A secondary login broker. When a database network includes more than one protocol, a secondary login broker performs broker functions for the second protocol.
Inactive	Placeholder in the server table for servers that have not yet started.

- **Protocol** — The communications protocol used by the server and its clients. TCP is the only valid value for this field.
- **Pend. Users** — The number of users who have successfully connected to the broker, but have not completed a connection to a server.
- **Cur. Users** — The number of users currently connect to a server.
- **Max. Users** — The maximum number of users who can connect to a server.

Processes/Clients

Displays status information about OpenEdge processes. The menu options allow you to choose all processes or a subset. Depending on your selection, the status display will contain a set of the columns described in the table below.

Table 90: PROMON Processes/Clients Status display columns

Column name	Description
Usr:Ten	The user number and tenant ID of the process.
Name	The user name of the process.
Domain	The domain name of the user of the process
Type	The process type. Table 80: Process type values on page 546 lists the possible types.
Wait	The wait type. Table 81: Possible wait column values on page 548 describes the possible types. If no value is displayed, the process is not waiting on any database-related events.
Wait Info	The ID of a resource the process is waiting on if the Wait field has a value. For example, if waiting on a record lock, Wait Info contains rowid : partition.
Login TimeStart Time	The time when the process logged into the database.
Trans id	The user's transaction ID.
Tx Start Time	The time the transaction started.
Trans State	The state of the transaction. Table 83: Transaction states on page 555 lists the possible states.
Pid	The process ID number.
Serv	The server number serving the remote client.
Schema Timestamp	The Unix Epoch timestamp of the user’s schema cache.

For certain wait values, an additional field is displayed. The table below lists the possible values.

Table 91: PROMON Processes/Clients Status Wait column additional values

Wait type	Additional information
REC	rowid

Wait type	Additional information
SCH	Type of schema lock: • 12 — share lock • 10 — exclusive lock
TRAN	taskid
RGET	rowid
BKSH	dbkey
BKEX	dbkey
BIRD	dbkey (bi)
BIWR	dbkey (bi)
AIRD	dbkey (ai)
ARWR	dbkey (ai)
TXS	2 — TXE update 3 — TEX commit
TXB	1 — TXE share 2 — TXE update 3 — TXE commit
TXE	1 — TXE share 2 — TXE update 3 — TEX commit
TXX	1 — TXE share 2 — TXE update 3 — TEX commit

The **Processes/Clients** menu choices are:

- **All Processes** — Displays status information about OpenEdge processes. The following shows a sample Processes/Clients Status display.

```
08/04/15        Status: All Processes by user number
18:06:31

  Usr:Ten   Name      Domain     Type       Wait   Wait Info  Trans id    Login time     Schema
 Timestamp
    0:0     dba                  BROK       --           0:0        0   08/04/15 15:23
          0
    5:4     lobtenant lobtenant  SELF/ABL   --           0:0        0   08/04/15 17:59
  1312480672
    6:2     tenant2   tenant2    SELF/ABL   --        4267:0        0   08/04/15 17:59
  1312480680
    9:0     dba                  MON        --           0:0        0   08/04/15 16:10
  1312474208
```

- **Blocked Clients** — The clients listed are waiting for a database-related resource. The following shows a sample Blocked Clients Status display. The clients listed are waiting for a database-related resource.

```
08/04/15        Status: Blocked Clients by user number
18:06:31

  Usr:Ten   Name      Domain     Type       Wait   Wait Info  Trans id    Login time     Schema
 Timestamp
    6:2     tenant2   tenant2    SELF/ABL   REC       4267:0        0   08/04/15 17:59
  1312480680
```

- **Active Transactions** — Displays status information about active transactions. The following shows a sample Active Transactions Status display.

```
08/04/15        Status: Active Transactions by user number
18:06:31

  Usr:Ten   Name      Domain     Type       Login time     Tx start time Trans id Trans State
   6:ST     st_barb        1    SELF/ABL    08/04/15 23:50 08/04/15 23:53    475      Active
```

- **Local Clients** — Displays status information about local client processes. The following shows a sample Local Clients status display.

```
08/17/15       Status: Local Clients by user number
19:21:09

Usr:Ten    Name      Domain   Type      Wait   Wait Info   Trans id  Login time        Pid
Schema Timestamp
  6       shannon             SELF/ABL   --                0         08/17/15 19:21  10664
     1318879261
```

- **Batch Clients** — Displays status information about local and remote batch client processes. The following shows a sample Local Batch Clients Status display.

```
09/16/15        Status: Batch Clients by user number for all tenants
15:18:12

Usr:Ten   Name      Domain    Type      Wait     Wait Info  Trans id   Login time    Pid
Schema Timestamp
  5       shannon       -4     SELF/ABL   --        0:0        29371  09/16/15 15:17 6451
    1438728061
```

- **Remote Clients** — Displays status information about remote client processes. The following shows a sample Remote Clients Status display.

```
09/09/15        Status: Remote Clients by user number for all tenants
20:20:02

Usr:Ten  Name     Domain   Type      Wait    Wait Info  Trans id   Login time    Serv IPV#
Remote Address  Schema Timestamp
 24      shannon      -3   REMC/ABL   --       0:0        29369  09/09/15 20:19    1 IPV4
192.168.197.28        1438728061
```

- **Background Processes** — Displays status information about background database processes. The following shows a sample Background Processes Status display.

```
08/25/15      Status: Background Processes Status
16:10

Usr:Ten   Name   Domain   Type    Start time          Pid
0:0       SYSTEM      0   BROK    08/25/15 10:31    16140
5:0       rgw         0   MON     08/25/15 11:12    16145
```

- **Connected Tenants** — Displays status information about connected tenants.

```
10/27/11            Currently Connected Tenants
00:14:21

     Sel     Tenant Id    Name             User Count
       1           -1     super_tenant           1
```

- **User Notification Processes** — Displays status information about users with Database Client Notification enabled.

```
09/26/16            Status: User Notification Processes
00:14:21

  Usr:Ten    Name       Domain      Type       Wait             Wait Info  Trans id  Poll
 time  Notifications   Last poll time    Schema Timestamp

    5        kmonico        -4       SELF/ABL   --                0:0             0
 12      R:-            09/26/16 10:12  1474899129
    8        kmonico        -4       SELF/ABL   --                0:0             0
 12      R:-            09/26/16 10:12  1474899132
   12        kmonico        -4       SELF/ABL   --                0:0             0
 12      R:-            09/26/16 10:12  1474899143
   16        kmonico        -4       SELF/ABL   --                0:0         34090
 12      -:R            09/26/16 10:12  1474899147
```

Files

Displays status information about OpenEdge database files.

PROMON Files Status display

The following shows a sample Files Status display.

```
09/28/06                Status: Files
12:15:05

                                        Size Extend
File name                               (KB) (KB)

/dev/dbs/docsample.db                    632  512
/dev/dbs/docsample.b1                   2168  512
/dev/dbs/docsample.d1                   1656  512
/dev/dbs/docsample_7.d1                  248  512
/dev/dbs/docsample_8.d1                 1144  512
/dev/dbs/docsample_9.d1                  248  512
/dev/dbs/docsample_10.d1                2040  512
/dev/dbs/docsample_11.d1                2040  512
/dev/dbs/docsample.a1                    504  0
/dev/dbs/docsample.a2                    504  0
/dev/dbs/docsample.a3                    120  512
/dev/dbs/docsample_1000.d1               632  0
/dev/dbs/docsample_1000.d2               120  512
/dev/dbs/docsample_2000.d1               632  0
/dev/dbs/docsample_2000.d2               120  512
```

The display contains the following fields:

- **File name** — The name of the file, including the path specification
- **Size** — The file size, in kilobytes
- **Extend** — The size of the last file extension, in kilobytes

Lock Table

Displays status information about the lock table. See Managing Performance on page 275 for a description of the lock table and its functions. When you choose this menu option, PROMON gives you five choices:

- Display all lock entries
- Display locks for one user
- Display locks for one ROWID
- Change list sorting
- Match a tenant or tenants

PROMON Lock Table Status display

The following figure shows a sample Lock Table Status display.

```
08/04/11        Status: Lock Table by user number
18:25:23
  Usr:Ten    Name       Domain      Trans id Type    Table:Part    ROWID Flags    Trans State
    6:2      tenant2    tenant2            0 REC         5:0         4290 S        None
```

The display contains the following fields:

- **Usr:Ten** — The user number and tenant ID of the user who owns the lock entry
- **Name** — For client processes, the user name; for a lock from a JTA transaction, `<JTA>` is displayed
- **Domain** — The domain name of the use
- **Trans id** — The transaction id
- **Type** — The lock type (for example, REC, RGET)
- **Table:Part** — The table number and partition ID
- **ROWID** — The ROWID of the record and lock table entry
- **Flags** — The table below lists the possible flags

Table 92: Lock table flags

Flag	Description
S	Share lock
X	Exclusive lock
U	Lock was upgraded from a share lock to an exclusive lock
L	Limbo—lock is being held until the transaction commits
Q	Request is queued, and the user is waiting for a conflicting lock held by another user

Flag	Description
P	Purged
H	Hold

- **Trans State** — Indicates the state of the transaction; see Table 83: Transaction states on page 555 for a description of the possible states

Buffer Cache

Displays status information about buffers.

PROMON Buffer Cache Status display

The following shows a sample Buffer Cache Status display.

```
09/25/11              Status: Buffer Cache
16:21

Total buffers:                  170
Hash Table Size:                43
Used Buffers:                   57
Empty Buffers:                  113
On lru Chain:                   168
On lru2 Chain:                  168
On apw queue:                   0
On ckp queue:                   0
Modified buffers:               170
Marked for ckp:                 0
Last checkpoint number:         0
LRU force skips:                0
LRU2 force skips:               0
```

The display contains the following fields:

- **Total buffers** — The total number of buffers in the buffer cache.
- **Hash Table Size** — The size of the buffer hash table.
- **Used Buffers** — The number of buffers currently in use.
- **Empty Buffers** — The number of empty buffers.
- **On lru Chain** — The number of buffers on the least recently used (LRU) chain.
- **On lru2 Chain** — The number of buffers on the least recently used (LRU) chain for the Alternate Buffer Pool (`-B2`).
- **On apw queue** — The number of buffers on the page writer queue.
- **On ckp queue** — The number of buffers on the checkpoint queue.
- **Modified buffers** — The number of dirty (modified) buffers.
- **Marked for ckp** — The number of buffers currently marked for checkpoint.
- **Last checkpoint number** — The most recent checkpoint number. As checkpoints begin, they are assigned a sequential number from the start of the session. The number is also the number of checkpoints that have occurred.

- **LRU force skips** — The number of times to access a buffer before moving it to the end of the LRU chain. (A value of 0 indicates this feature is disabled.)
- **LRU2 force skips** — The number of times to access a buffer in the alternate buffer pool before moving it to the end of the LRU chain. (A value of 0 indicates this feature is disabled.)

Logging Summary

Displays status information for the database logging function.

PROMON Logging Summary Status display

The following shows a sample Logging Summary Status display.

```
09/25/06        Status: Logging Summary
16:22

Crash protection:                Yes
Delayed Commit:                  3 seconds
Before-image I/O:                Reliable
Before-image cluster age time:   0
BI Writer status:                Executing
Two-Phase Commit:                *** OFF ***
After-image journalling:         Yes
After-image I/O:                 Reliable
AI Writer status:                Not executing
```

The display contains the following fields:

- **Crash protection** — Yes, if crash protection is enabled; no, if it is not.
- **Delayed Commit** — The current value of the Delayed BI File Write (`-Mf`) parameter.
- **Before-image I/O** — One of the following values:
  - Reliable if synchronous or direct I/O is being used for BI writes
  - BUFFERED if buffered I/O is being used for BI writes. Buffered I/O is not recommended
- **Before-image cluster age time** — The period of time that must pass before a BI cluster is reused. This period ensures that database blocks flushed at checkpoint are moved from the UNIX buffers on disk. When this occurs, the transaction is recorded on disk.
- **BI Writer status** — Executing, if a BIW is running.
- **Two-Phase Commit** — ON, if two-phase commit is enabled; OFF, if it is not.
- **After-image journalling** — ON, if after-imaging is enabled; OFF, if it is not.
- **After-image I/O** — One of the following values:
  - Reliable if synchronous or direct I/O is being used for AI writes
  - BUFFERED if buffered I/O is being used for AI writes. Buffered I/O is not recommended
- **AI Writer status** — Executing, if an AIW is running.

BI Log

Displays status information for before-image logging.

PROMON BI Log Status display

The following shows a sample BI Log Status display.

```
09/25/16       Status:BI Log
16:23

Before-image cluster age time:                  60 seconds
Before-image block size:                        8192 bytes
Before-image cluster size:                      512 kb (524288 bytes)
Number of before-image extents:                 1
Before-image log size (kb):                     2168
Bytes free in current cluster:                  521177 (100%)
Last checkpoint was at:                         09/25/16 13:56
Number of BI buffers:                           5
Full buffers:                                   0
Current BI Cluster:                             4
BI Cluster HWM:                                 4
```

The display contains the following fields:

- **Before-image cluster age time** — The period of time that must pass before a BI cluster is reused. This period ensures that database blocks flushed at checkpoint are moved from the buffers on disk. When this occurs, the transaction is durably recorded on disk.
- **Before-image block size** — The BI block size.
- **Before-image cluster size** — The BI cluster size.
- **Number of before-image extents** — The number of BI files.
- **Before-image log size (kb)** — The size of the BI log file, in kilobytes.
- **Bytes free in current cluster** — The number of free bytes remaining in the current BI cluster.
- **Last checkpoint was at** — The time of the last checkpoint.
- **Number of BI buffers** — The number of BI buffers.
- **Full buffers** — The number of full BI buffers.
- **Current BI Cluster** — The number of the current BI cluster.
- **BI Cluster HWM** — The high water mark of the BI clusters.

AI Log

Displays status information for after-image logging, if after-imaging is enabled.

PROMON AI Log Status display

The following shows a sample AI Log Status display.

```
09/15/16        Status: AI Log
11:35:24

After-image begin date:            08/18/16 10:47
After-image new date:              08/18/16 10:47
After-image open date:             09/15/16 11:31
After-image generation number:     1
Number of after-image extents:     3
Current after-image extent:        13
Number of AI buffers:              5
After-image block size:            8192 bytes
After-image log size:              2168 K
After-image sequence #:            1
```

The display contains the following fields:

- **After-image begin date** — The date of the last AIMAGE BEGIN command.
- **After-image new date** — The date of the last AIMAGE NEW command.
- **After-image open date** — The date when the AI log was last opened.
- **After-image generation number** — The generation number of the current AI extent.
- **Number of after-image extents** — The number of AI files or extents.
- **Current after-image extent** — The extent number of the current (busy) extent.
- **Number of AI buffers** — The number of buffers in the AI buffer pool. You can change the number of AI buffers with the After-image Buffers (`-aibufs`) startup parameter. The default value is 1.
- **After-image block size** — The size of the after-image block. You can change the AI block size. This reduces I/O rates on disks where AI files are located. See Managing Performance on page 275 for more information.
- **After-image log size** — The size of current busy AI extent, in kilobytes.
- **After-image sequence #** — The current AI sequence number.

Two-phase Commit

Displays status information about two-phase commit, if enabled.

PROMON Two-phase Commit Status display

The following shows a sample Two-phase Commit Status display.

```
09/25/03            Status: Two-Phase Commit
17:02:19
Coordinator nickname:     mydemo
Coordinator priority:     0
After-image journalling:  Yes
```

The display contains the following fields:

- **Coordinator nickname** — The nickname used to identify the coordinator database. You specify a nickname when you enable two-phase commit.

- **Coordinator priority** — The priority for the coordinator database. You specify the priority when you enable two-phase commit.
- **After-image journalling** — Yes, if after-imaging is enabled; no, if it is not.

Startup Parameters

Displays values of OpenEdge database startup parameters.

PROMON Startup Parameters Status display

The following shows a sample Startup Parameters Status display.

```
03/07/17        Status: Startup Parameters
20:15:46

              After-image Management Archival Directory List (-aiarcdir): Not Enabled
   Create After-image Management Archival Directory(s) (-aiarcdircreate): Not Enabled
               After-image Management Archival Interval (-aiarcinterval): -1
                                 Number of After-Image Buffers (-aibufs): 20
                                        After-Image Stall (-aistall): Not Enabled
                  Starting index number for statistics range (-baseindex): 1
                  Starting table number for statistics range (-basetable): 1
    Starting index number per user for statistics range (-baseuserindex): 1
    Starting table number per user for statistics range (-baseusertable): 1
                                Number of Before-Image Buffers (-bibufs): 20
                                      BI File Threshold Stall (-bistall): Disabled.
                                       BI File Threshold size (-bithold): 0.0   Bytes
                                         Database Blocksize (-blocksize): 4096
                                             BIW writer delay (-bwdelay): 0
                                              Allowed index cursors (-c): 84
                                               CDC cache size (-cdcsize): 200k bytes
                             SSL Certificate Store Path (-certstorepath): Not Enabled
                                             Character Set (-cpinternal): iso8859-1
                                           Physical Database Name (-db): jan28
                                        Diagnostic directory (-diagDir): Not Enabled
      Diagnostic events value (-diagEvent): LockTable:0,BiThold:0,SysErr:0
                                  Diagnostic event level (-diagEvtLevel): 0
                                    Diagnostic field separator (-diagFS): ' '
                                    Diagnostic data format (-diagFormat): csv
                                    Diagnostic pause length (-diagPause): 0
                                   Diagnostic prefix value (-diagPrefix): diagEvent_
                                        Diagnostic procedure (-diagProc): Not Enabled
                                                  Direct I/O (-directio): Not Enabled
                                                     Database Type (-dt): PROGRESS
                                      Encryption cache size (-ecsize): 1000
                                            Group delay (-groupdelay): 10
                                          Hash Table Entries (-hash): 887
                               Buffer hash latch factor (-hashLatchFactor): 10
                                                   Crash Recovery (-i): Enabled
   Number of indexes included in statistics collection (-indexrangesize): 50
                                               TCP/IP Version (-ipver): Not Enabled
                                        SSL Key Alias Name (-keyalias): Not Enabled
                                     Lock table hash table size (-lkhash): 1237
                                  Original Lock Release Algorithm (-lkrela): Not Enabled
                                      Number of LRU force skips (-lruskips): 0
                                    Number of LRU2 force skips (-lru2skips): 0
                                         Maximum Area Number (-maxArea): 32000
                                  Size of JTA transaction table (-maxxids):  100
                                  Maximum Port for Auto Servers (-maxport): 0
                                  Minimum Port for Auto Servers (-minport): 0
                           Multi-tenancy partition cache size (-mtpmsize): 1024
                                                Use muxlatches (-mux): 1
                                       Maximum Number of Users (-n): 21
                     Minimum time to nap at first -spin exhaustion (-nap): 10
```

```
                    Maximum time to nap at -spin exhaustion (-napmax): 250
                                  SSL No Host Verify (-nohostverify): Not Enabled
                                      Disable LRU mechanism (-nolru): Not Enabled
                                SSL No Session Cache (-nosessioncache): Not Enabled
                                SSL No Session Reuse (-nosessionreuse): Not Enabled
                                        Login Governor (-nGovernor): 0 of 21
      Number of checkpoint statistics to record (-numCheckpointStats): 32
                                Storage object cache size (-omsize): 1024
                  Database Service Manager - IPC Queue Size (-pica): 64.0  KBytes
                             Database PICA queue nap time (-picanap): 1 ms
                             Shared memory segments locked (-pinshm): Not Enabled
                  Use pollset mechanism for client/server (-pollset): Not Enabled
                       Delay first prefetch message (-prefetchDelay): Not Enabled
                  Prefetch message fill percentage (-prefetchFactor): 0
                  Minimum records in prefetch msg (-prefetchNumRecs): 16
                  Suspension queue poll priority (-prefetchPriority): 0
               APW queue scan cycle time in milliseconds (-pwqdelay): 100
                     APW minimum queue length before write (-pwqmin): 1
                APW buffer scan cycle time in seconds (-pwsdelay): 1
          APW maximum number of buffers to scan per cycle (-pwscan): 5
         APW maximum number of buffers to write per cycle (-pwwmax): 25
                                    Before-Image File I/O (-r -R): Reliable
        Record free chain search depth factor (-recspacesearchdepth): 5
                                    Security cache size (-secsize): 512
                                Number of Semaphore Sets (-semsets): 3
                             SSL Session Timeout (-sessiontimeout): 0
                  Maximum Shared Memory Segment Size (-shmsegsize): 0 Mb
                                Current Spin Lock Tries (-spin): 24000
                        SSL Encryption for TCP/IP connections (-ssl): Not Enabled
                  STS Debug Logging Level default (-stslogginglevel): 0
  Number of tables included in statistics collection (-tablerangesize): 50
                           User Notification Time (-usernotifytime): 0 seconds
Number of indexes per user included in statistics collection (-userindexrangesize): 50
Number of tables per user included in statistics collection (-usertablerangesize): 50
                         Area block consistency check (-AreaCheck): Not Enabled
                                  Number of Database Buffers (-B): 3000
                        Number of Alternate Database Buffers (-B2): 0
                         Maximum private buffers per user (-Bpmax): 64
                        Database block consistency check (-DbCheck): Not Enabled
        Database Service Manager - Service(s) to start (-DBService): Not Enabled
                                    Enhanced Read-Only mode (-ERO): Not Enabled
                                            Force Access (-F): Not Enabled
                              Before-Image Truncate Interval (-G): 0
                                               Host Name (-H): Not Enabled
                      Index block consistency check (-IndexCheck): Not Enabled
                                  Current Size of Lock Table (-L): 8192
                                       Lock Governor (-LGovernor): 0%
                       Maximum Number of Clients Per Server (-Ma): 4
                              Memory overwrite check (-MemCheck): Not Enabled
                                Delay of Before-Image Flush (-Mf): 3
                                 Minimum Clients Per Server (-Mi): 0
                                        Message Buffer Size (-Mm): 0
                                Maximum Number of Servers (-Mn): 5
                                     Servers per Protocol (-Mp): 0
                                Maximum Servers Per Broker (-Mpb): 0
                                 Excess Shared Memory Size (-Mxs): 26
                                            Network Type (-N):
                    Server network message wait time (-Nmsgwait): 2
           Pending client connection timeout (-PendConnTimeout): 0
                                            Service Name (-S): 0
                       Broker server group support (-ServerType): BOTH
                       SQL Server Max Open Cursors (-SQLCursors): 0
                                SQL Server Stack Size (-SQLStack): 0
                  SQL Server Statement Cache Size (-SQLStmtCache): 0
Size [1K byte units] of SQL Server temp table buffer (-SQLTempStoreBuff): 0
Size [1K byte units] of SQL Server temp table disk storage (-SQLTempStoreDisk): 0
Size [1K byte units] of SQL Server temp table data page (-SQLTempStorePageSize): 0
                  Authorized data truncation (-SQLTruncateTooLarge): OFF
                     SQL Autonomous Schema Update (-SQLWidthUpdate): OFF
```

```
                        Record block consistency check (-TableCheck): Not Enabled
                              TXE Lock retry limit (-TXERetryLimit): 0
                          TXE Commit lock skip limit (-TXESkipLimit): 10000
```

The display contains every startup parameter in alphabetical order. See Database Startup Parameters on page 473 .

Shared Resources

Displays status information about shared database resources.

PROMON Shared Resources Status display

The following shows a sample Shared Resources Status display.

```
10/08/14        Status: Shared Resources
17:27:37

Active transactions:              0
Lock table entries in use:        0 of 8192
Lock table high water mark:       1
Primary object cache (-omsize):   Total: 1024 Used: 237 Pinned: 0
Secondary object cache (-omsize): Total: 1024 Used: 0 Pinned: 0
Primary MT cache (-mtpmsize):     Total: 1024 Used: 0 Pinned: 0
Secondary MT cache (-mtpmsize):   Total: 1024 Used: 0 Pinned: 0
Number of servers:                3 (5 allocated)
Total users:                      0 (20 allocated)
   Self-service:                  0
   Remote:                        0
   Batch:                         0
Watchdog status:                  *** Not executing ***
BIW status:                       *** Not executing ***
Number of page writers:           0
Number of monitors:               1
Number of semaphores allocated:   29
Shared memory allocated:           17671K (1 segments. The last segment was not
 locked in memory)
```

The display contains the following fields:

- **Active transactions** — The number of currently active transactions
- **Lock table entries in use** — The number of lock table entries that are currently being used, and the total number of entries available
- **Lock table high water mark** — The maximum number of lock table entries that were in use simultaneously
- **Primary object cache** — The number of entries in the primary storage object cache
- **Secondary object cache —** The number of entries in the secondary storage object cache; This number is identical to the primary unless it has been increased by PROUTIL INCREASETO
- **Primary MT cache —** The number of entries in the primary multi-tenancy partition cache
- **Secondary MT cache —** The number of entries in the secondary multi-tenancy partition cache. storage object cache; This number is identical to the primary unless it has been increased by PROUTIL INCREASETO

- **Number of servers** — The number of servers running
- **Total users** — The number of users, broken down by the following types: Self service, Remote, and Batch
- **Watchdog status** — Executing, if the Watchdog process is executing. Not executing, if the Watchdog is not executing
- **BIW status** — Executing, if the BIW is executing. Not executing, if the BIW is not executing
- **AIW status** — Executing, if the AIW is executing. Not executing, if the AIW is not executing
- **Number of page writers** — The number of APWs running
- **Number of monitors** — The number of Monitor processes running
- **Number of semaphores allocated** — The number of UNIX semaphores allocated
- **Shared memory allocated** — The amount of shared memory allocated, in kilobytes and in segments. See Managing Performance on page 275 for more information on shared-memory allocation.

Shared Memory Segments

Displays general information about shared-memory segments allocated to the database.

PROMON Shared Memory Segments Status display

The following shows a sample Shared Memory Segments Status display.

```
09/20/06        Status: Shared Memory Segments
13:17:44

Seg     Id      Size          Used       Free
 1      91242   858972160  849424036    9548124
```

The display contains the following fields:

- **Seg** — The segment number; each segment receives a sequential number when it is created
- **Id** — The number the operating system uses to reference this segment
- **Size** — The total allocated segment size, in bytes
- **Used** — The amount of shared memory used from the segment, in bytes
- **Free** — The amount of shared memory allocated but not yet used in the segment, in bytes

AI Extents

Displays the status of the database AI extents, if after-imaging is enabled.

PROMON AI Extents status display

The following shows a sample AI Extents status display.

```
09/15/05        Status: AI Extents
11:42:07

   Area  Status     Type     File      Size     Extent Name
                             Number   (KBytes)
     13  BUSY       Fix           1        505   /usr1/101A/doc_sample.a1
     14  EMPTY      Fix           0        505   /usr1/101A/doc_sample.a2
     15  EMPTY      Var           0        121   /usr1/101A/doc_sample.a3
```

The display contains the following fields:

- **Area** — The area number of the extent. Each AI extent has a unique area number
- **Status** — The status of the extent. Possible values are:
  - BUSY
  - EMPTY
  - FULL
  - LOCKED
  - ARCHIVED
- **Type** — The extent type:
  - Fixed length
  - Variable
- **File Number** — After image sequence number
- **Size** — The size of the extent in kilobytes
- **Extent Name** — The exact file name the operating system uses to reference this extent

Database Service Manager

Displays the status of communications activity between the core database and related database service objects.

```
09/29/11        Status: Database Service Manager
10:51:13

Communication Area Size   :        65.00 KB
  Total Message Entries   :          595
  Free Message Entries    :          595
  Used Message Entries    :            0
  Used HighWater Mark     :            0
  Area Filled Count       :            0
  Service Latch Holder    :           -1
  Access Count            :            0
  Access Collisions       :            0

Registered Database Service Objects

Name                            Rdy Status  Messages Locked by
OpenEdge Replication Server      Y  BUSYNP         1
OpenEdge RDBMS                   Y  REG            0
```

The display contains the following fields:

- **Communication Area Size** — The size in KB of the communications queue
- **Total Message Entries** — The total number of messages that can be queued
- **Free Message Entries** — Number of free messages
- **Used Message Entries** — Number of used messages
- **Used HighWater Mark** — The maximum number of PICA messages used for this run of the database
- **Area Filled Count** — The number of times the PICA queue has been filled for this run of the database
- **Service Latch Holder** — User number of the user holding the database service manager queue (PICA) latch; if the value displayed is -1, then the latch is not currently held/locked/owned
- **Access Count** — The number of times the PICA queue has been accessed for this run of the database
- **Access Collisions** — The number of times that there has been collision on access to the PICA queue during this run of the database (AIW and REPL attempting simultaneous access)
- **Registered Database Service Objects**:
  - **Name**— The database service name; possible values include:
    - OpenEdge Replication Server
    - OpenEdge Replication Agent
    - OpenEdge RDBMS
    - OpenEdge Management DB Agent
- **Rdy** — **Y** or **N**. Indicates if the service is ready. The value is most often **Y**. **N** is displayed during service startup and shutdown.
- **Status** — The status of the database service

  The table below lists the possible values for **Status**.

Table 93: Status values

Value	Description
RUN	Running
ACT	Active — the process exists; it is registered and the broker is aware
INACT	Inactive — the service is registered, but not processing or available
BUSY	Busy — processing
REG	Registered — the service is known, but has no role
PREREG	Pre-registered. The service is known, but it not ready for use (starting registration)
UNK	Unknown

- **Messages** — The number of messages on the database service queue
- **Locked by** — The pid of the process locking the message queue

Servers by broker

Displays the number of servers for each broker.

```
12/10/07        Status: Servers By Broker
09:05:46

Sv                                                  Pend.   Cur.   Max.   Port
No    Pid  Type       Protocol              Logins  Users  Users  Users    Num
```

The display contains the following fields:

- **Service number** — Indicates the service number
- **Pid** — The process ID number
- **Type** — The process type — Table 80: Process type values on page 546 lists the possible types
- **Protocol** — The protocol type
- **Logins** — The number of logins
- **Pending Users** — The number of pending users
- **Current Users** — The number of current users
- **Maximum Users** — The maximum number of users
- **Port Number** — The port number

Client Database-Request Statement Cache

Displays options for Client Database-Request Statement Caching.

```
04/08/15        OpenEdge Release 11 Monitor (R&D)
09:06:10        Client Database-Request Statement Caching Menu

                 1. Activate For Selected Users
                 2. Activate For All Users
                 3. Activate For All Future Users
                 4. Deactivate For Selected Users
                 5. Deactivate For All Users
                 6. Deactivate For All Future Users
                 7. View Database-Request Statement Cache
                 8. Specify Directory for Statement Cache Files
                 9. Toggle between User and Tenant Options
                10. Database statement cache locks
```

Choose one of the following options:

- **Activate For Selected Users** — Activates Client Database-Request Statement Caching for one or more clients. For more information, see Activating database-request statement caching for selected users on page 588.
- **Activate For All Users** — Activates Client Database-Request Statement Caching for all clients. For more information, see Activating database-request statement caching for all users on page 589.
- **Activate For All Future Users** — Activates Client Database-Request Statement Caching for all future client connections to the database. For more information, see Activating database-request statement caching for all future users on page 589.
- **Deactivate For Selected Users** — Deactivates Client Database-Request Statement Caching for one or more clients. For more information, see Deactivating database-request statement caching for selected users on page 589.
- **Deactivate For All Users** — Deactivates Client Database-Request Statement Caching for all clients.
- **Deactivate For All Future Users** — Deactivates Client Database-Request Statement Caching for all future client connections to the database.
- **View Database-Request Statement Cache** — View the Client Database-Request Statement Cache. For more information, see Viewing database-request statement caching on page 590.
- **Specify Directory for Statement Cache Files** — Specify the directory for all temporary database-request statement cache files. For more information, see Specifying a directory for database-request statement caching files on page 591.
- **Toggle between User and Tenant Options** — When selected, switches options 1-6 to activate or deactivate based on tenancy not on users.
- **Database statement cache locks** — When selected, displays a list of locks obtained by users performing database statement caching.

Activating database-request statement caching for selected users

When you choose to activate database-request statement caching for an individual user, the following menu appears:

```
Enter the type of statement caching to activate (1-Single, 2-Stack, 3-One Time Request,
 Q-Quit):
```

Enter the number for the type, or Q to exit:

- **1-Single** — Only the current ABL program and line number, or a single SQL statement is reported.
- **2-Stack** — The call stack up to the current ABL program and line number.
- **3-One Time Request** — The ABL or SQL client report tracing information once. Once reported, database-request statement caching is turned off.
- **Q-Quit** — Quit this function and return to the Client Database-Request Statement Caching menu.

After selecting the type, chose a user by number from a menu like the following:

```
Enter the type of statement caching to activate (1-Single, 2-Stack, 3-One Time Request,
 Q-Quit): 2

04/08/15        OpenEdge Release 11 Monitor (R&D)
12:23:14        Activate For Selected Users

Usr:Ten Name    Domain   Type       Login time     Serv IPV# Remote Address
  5     User-5      -4   SELF/ABL   04/08/15 12:23    0
  6     User-6      -4   SELF/ABL   04/08/15 12:23    0
  7     User-7      -4   SELF/ABL   04/08/15 12:23    0
  8     User-8      -4   SELF/ABL   04/08/15 12:23    0

Enter user number, P, T, or X (? for help):
```

Based on your entry, the following actions are performed:

- If the user number is valid, or if the user is specified in the list, database-request statement caching is activated. If the user is a self-serve or remote client, database-request statement caching is activated for that client. If the user number represents a remote database server, database-request statement caching is activated for all clients serviced by that database server.

  Note: Once activated, database-request statement caching can not be canceled. You must deactivate for selected users, all users, or all future users.

- You can continue entering additional valid user numbers.
- The user number entry is terminated when you press ENTER without specifying a valid numeric character, or when you press ENTER after entering a non-numeric character.

Activating database-request statement caching for all users

When you choose to activate database-request statement caching for all users, the following menu appears:

```
Enter the type of statement caching to activate (1-Single, 2-Stack, 3-One Time Request,
 Q-Quit):
```

Enter the number for the type, or Q to exit:

- **1-Single** — Only the current ABL program and line number, or a single SQL statement is reported by the ABL client.
- **2-Stack** — The call stack up to the current ABL program and line number.
- **3-One Time Request** — The ABL or SQL client report tracing information once. Once reported, database-request statement caching is turned off.
- **Q-Quit** — Quit this function and return to the Client Database-Request Statement Caching menu.

Once the type of database-request statement caching is selected, tracing is activated for all clients currently connected to the database.

Note: Once activated, database-request statement caching remains activated until deactivated.

Activating database-request statement caching for all future users

When you choose to activate database-request statement caching for all future users, the following menu appears:

```
Enter the type of statement caching to activate (1-Single, 2-Stack,Q-Quit):
```

Enter the number for the type, or Q to exit:

- **1-Single** — Only the current ABL program and line number, or a single SQL statement is reported by the ABL client.
- **2-Stack** — The call stack up to the current ABL program and line number.
- **Q-Quit** — Quit this function and return to the Client Database-Request Statement Caching menu.

Once the type of database-request statement caching is selected, tracing is activated for all future users.

Note: Once activated, database-request statement caching remains activated until deactivated.

Deactivating database-request statement caching for selected users

Once database-request statement caching is activated for one or more users, any deactivate menu option will halt database-request statement caching.

Note: To view the database-request statement cache, select **View Database-Request Statement Cache** from the main **Client Database-Request Statement Caching** menu. For more information, see Viewing database-request statement caching on page 590.

When you choose to deactivate database-request statement caching for selected users, a list of users with database-request statement caching activated is generated, like the following:

```
04/08/15        OpenEdge Release 11 Monitor (R&D)
13:12:30        Deactivate For Selected Users

Usr:Ten Name    Domain   Type       Login time     Serv IPV# Remote Address
  5     User-5      -4   SELF/ABL   04/08/15 13:33    0
  6     User-6      -4   SELF/ABL   04/08/15 13:34    0
  7     User-7      -4   SELF/ABL   04/08/15 13:34    0
  8     User-8      -4   SELF/ABL   04/08/15 13:34    0

Enter user number, P, T, or X (? for help):
```

Once the user information is displayed, you can deactivate database-request statement caching for selected users by specifying the corresponding user number. Based on the user number, the following actions are performed:

- If the user number is valid, and if the user is specified in the list, database-request statement caching is deactivated. If the user is a self-serve or remote client, database-request statement caching is deactivated for that client. If the user number represents a remote database server, database-request statement caching is deactivated for all clients serviced by that database server.
- You can continue entering additional valid user numbers.
- The user number entry is terminated when you press ENTER without specifying a valid numeric character, or when you press ENTER after entering a non-numeric character.

Viewing database-request statement caching

When you choose to view the database-request statement cache, the following menu appears:

```
04/08/15        OpenEdge Release 11 Monitor (R&D)
13:38:46        View Client Statement Cache

Usr:Ten Name    Domain   Type       Login time     Serv IPV# Remote Address
  5     User-5      -4   SELF/ABL   04/08/15 12:23    0
  6     User-6      -4   SELF/ABL   04/08/15 12:23    0
  7     User-7      -4   SELF/ABL   04/08/15 12:23    0
  8     User-8      -4   SELF/ABL   04/08/15 12:23    0

Enter user number, P, T, or X (? for help): 6

04/08/15        OpenEdge Release 11 Monitor (R&D)
13:39:02        View Client Statement Cache

User number                    : 6
User name                      : User-6
User type                      : SELF/ABL
Login date/time                : 04/08/15 12:23

Statement caching type         : SQL Statement or Single ABL Program Name
Statement caching last updated : 04/08/15 14:11

Statement cache                :  -1 : d_mar18.p
                                  2 : /scratch/doc/116/p94309_Untitled1.ped


Enter user number, P, T, or X (? for help):
```

The following table provides information about fields in the **View Client Statement Cache** display.

Table 94: View client statement cache display fields

Field	Description
User number	The user number associated with the database connection.
Name	The user name associated with the connected user.
User Type	The type of database connection.
Login date/time	The date and time the user connected to the database.
Statement cache type	**SQL statement or Single ABL Program Name** — The statement cache information contains a single SQL statement or a partial ABL program line number. **SQL statement or Partial ABL Program Name** — The statement cache information contains a single SQL statement or a partial ABL program stack containing ABL program names and line numbers. **One Time SQL Statement or Partial ABL Program Name** — The statement cache information contains a one time request of a single SQL statement or a partial ABL program stack containing ABL program names and line numbers.
Statement cache last updated	The date and time the database-request statement cache was updated.
Statement cache	The SQL statement or ABL program information for the last database request made by the client. This field may change, if: • The client is an ABL client, the line number is shown followed by the ABL program name. The line number displayed is the one generated by the compiled listing of the ABL program; it may not match the contents of the ABL source code. • The client is a SQL client; the SQL statement is displayed.

Specifying a directory for database-request statement caching files

If the database-request statement cache information for a client exceeds 256 bytes of shared memory it is stored in a temporary file. You must specify a directory for these temporary files.

To specify a directory to store database-request statement cache files, choose **option 8** in the **Client Database-Request Statement Caching** menu. The following appears:

```
04/08/15        Status: Specify Directory for Temporary Files
15:49:15

Enter the name of the statement Cache Temporary Directory:
```

Enter the name of the directory. The directory name can not exceed 255 bytes in length, and must be created prior to displaying accessing this option.

Note: If the directory name is not specified, and the database-request statement cache information is greater than 256 bytes, the information is written to a file in the current working directory.

Schema Locks & Wait Queue

Displays outstanding schema locks.

```
10/26/11        Status: Schema Locks & Wait Queue by user number
23:39:36

  Usr:Ten    Name      Domain    Locked     Queued

**** No schema locks outstanding.
```

Broker Startup Parameters

Displays startup parameters specific to brokers (primary and secondary).

```
10/08/14        Status: Broker Startup Parameters
16:49:52

Broker: 0 Pid: 8988 Logins: 0 Pend: 0 Connected: 0
                                Maximum Port for Auto Servers (-maxport): 5000
                                Minimum Port for Auto Servers (-minport): 3000
                              Maximum Number of Clients Per Server (-Ma): 5
                                        Minimum Clients Per Server (-Mi): 1
                                               Message Buffer Size (-Mm): 1024
                                             Servers per Protocol (-Mp): 0
                                       Maximum Servers Per Broker (-Mpb): 0
                                                   Network Type (-N): TCP
                     Pending client connection timeout (-PendConnTimeout): 0
                                                   Service Name (-S): 1234
                                Broker server group support (-ServerType): BOTH
                                SQL Server Max Open Cursors (-SQLCursors): 0
                                        SQL Server Stack Size (-SQLStack): 0
                           SQL Server Statement Cache Size (-SQLStmtCache): 0
Size [1K byte units] of SQL Server temp table buffer (-SQLTempStoreBuff): 0
Size [1K byte units] of SQL Server temp table disk storage (-SQLTempStoreDisk):
0
Size [1K byte units] of SQL Server temp table data page (-SQLTempStorePageSize)
 0

Enter <return>, R, P, T, or X (? for help):
```

R&D Activity Displays

The R&D Activity displays provide tables of general information regarding database activity. Activity can be queried for the following:

- Summary on page 594
- Servers on page 569
- Buffer Cache on page 576
- Page Writers on page 598
- BI Log on page 577
- AI Log on page 578

- Lock Table on page 575
- I/O Operations By Type on page 602
- I/O Operations by File on page 603
- Space Allocation on page 604
- Index on page 605
- Record on page 605
- Other on page 606

The pages of activity displays end with this prompt:

```
Enter <return>, A, L, R, S, U, Z, P, T, or X (? for help):
```

The table below describes the actions PROMON takes, based on the input received.

Table 95: Activity display prompt

If you type . . .	PROMON will . . .
ENTER	Display the next page of the current display if one exists, otherwise return to the previous menu.
A	Activate auto-repeat mode. The current display restarts after the number of seconds specified by the current display pause time. Auto-repeat mode stops after the number of times specified by the auto repeat count or when **CTRL+C** is pressed.
L	Load activity counters from start of session. Use this option to reset the activity counters after a sampling a smaller time slice.
R	Repeat the current display.
S	Sample activity counters for the number of seconds specified by the sampling interval. The current display then restarts. The data shown include only activity which occurred during the sample interval.
U	Update activity counters. The current display then restarts. The data shown include changes which occurred since the initial set of data was collected.
Z	Zero the activity counters so updates show changes from this time forward.
P	Return to the previous menu.
T	Return to the top-level (main) menu.
X	Exit from the OpenEdge Monitor utility.
?	Display help about the entry choices.

Summary

Displays general information about database activity.

PROMON Summary Activity display

The following shows a sample Summary Activity display.

```
09/25/03       Activity: Summary
16:27          from 09/25/03 15:31 to 09/25/03 16:04 (2 min. 48 sec.)

Event           Total   Per Sec.    |Event          Total   Per Sec.
Commits          1208        0.7    |DB Reads         57        0.0
Undos               0        0.0    |DB Writes         3        0.0
Record Reads     1334        0.8    |BI Reads         13        0.0
Record Updates   1208        0.7    |BI Writes       131        0.0
Record Creates      0        0.0    |AI Writes         0        0.0
Record Deletes      0        0.0    |Checkpoints       0        0.0
Record Locks     3625        2.3    |Flushed at chkpt  0        0.0
Record Waits        0        0.0     Active trans:     1
Rec Lock Waits  0%    BI Buf Waits      0%    AI Buf Waits      0%
Writes by APW   0%    Writes by BIw    96%    Writes by AIW     0%
DB Size:      352K    BI size:        512K    AI Size:          0K
Empty blocks:   10    Free blocks:       0    RM chain:         1
Buffer Hits     99%   Primary Hits      99%   Alternate Hits   99%
0 Servers, 1 Users (1 Local, 0 Remote, 0 Batch), 1 APWs
```

The Summary Activity display shows the number of events that have occurred on the system, including the cumulative total and the number of events per second. The display includes the following events:

- **Commits** — The number of transactions all users have committed
- **Undos** — The number of transactions rolled back
- **Record Reads** — The number of records read
- **Record Updates** — The number of records updated
- **Record Creates** — The number of records created
- **Record Deletes** — The number of records deleted
- **Record Locks** — The number of record locks used
- **Record Waits** — The number of times users have waited to access a locked record
- **DB Reads** — The number of database blocks read
- **DB Writes** — The number of database blocks written to disk
- **BI Reads** — The number of BI blocks read
- **BI Writes** — The number of BI blocks written to disk
- **AI Writes** — The number of AI blocks written to disk
- **Checkpoints** — The number of checkpoints that have been performed
- **Flushed at chkpt** — The number of database buffers that have been flushed to disk because they were not written by the time the checkpoint ended
- **Active trans** — The number of active transactions

The Summary Activity display also shows the following information:

- **Rec Lock Waits** — The percentage of record accesses that result in record lock waits. A record lock wait occurs when the database engine must wait to access a locked record.

  For optimum performance, try to keep this number as low as possible. You can lower this number by modifying your application to perform shorter transactions so that record locks are held for shorter periods of time.

- **BI Buf Waits** — The percentage of BI buffer waits. A BI buffer wait occurs when the database engine must wait to access a BI buffer.

  For optimum performance, try to keep this number as low as possible. To decrease this number, allocate more BI buffers with the Before-image Buffers (`-bibufs`) parameter.

- **AI Buf Waits** — The percentage of AI buffer waits. An AI buffer wait occurs when the database engine must wait to access an AI buffer.

  For optimum performance, try to keep this number as low as possible. To decrease this number, allocate more AI buffers with the After-image Buffers (`-aibufs`) parameter.

- **Writes by APW** — The percentage of database blocks written to disk by the APW; this is a percentage of the total number of database blocks written.

  For optimum performance, try to keep this number as high as possible. To increase this number, start more APWs and increase the cluster size.

- **Writes by BIW** — The percentage of BI blocks written to disk by the BIW; this is a percentage of the total number of BI blocks written to disk. For optimum performance, try to keep this percentage fairly high. You can increase the number of BI buffers with the `-bibufs` parameter.

- **Writes by AIW** — The percentage of AI blocks written to disk by the AIW; this is a percentage of the total number of AI blocks written. For optimum performance, try to keep this percentage fairly high. You can increase the number of AI buffers with the `-aibufs` parameter.

- **DB Size** — The size of your database, in kilobytes.

- **BI Size** — The size of your BI file, in kilobytes.

- **AI Size** — The size of your AI file, in kilobytes.

- **Empty blocks** — The number of empty (never used) database blocks.

- **Free blocks** — The number of blocks on the database's free chain. The free chain is a chain of previously used and then deallocated database blocks.

- **RM chain** — The number of blocks on the database's RM chain. The RM chain is a chain of partially filled database blocks.

- **Buffer Hits** — The percentage of buffer hits for both the primary and Alternate Buffer pools. A buffer hit occurs when the database engine locates a database record in the buffer pool and does not have to read the record from disk.

  For optimum performance, keep this number as high as possible. To increase this number, allocate more buffers with the `-B` and `-B2` parameters. Increase the number of buffers until one of the following occurs:

  - The number of I/Os per second is reduced to less than half the capacity of the drives
  - The buffer hit rate stops improving
  - The system starts paging

- **Primary Hits** — The percentage of buffer hits for the primary buffer pool.

- **Alternate Hits** — The percentage of buffer hits for the Alternate Buffer Pools.

The last line of the Summary Activity display summarizes the current number of each type of process running against the database at the time you run the Activity option. The table below defines the process types.

Table 96: Process types

Field	Description
Servers	Number of servers running against your database. This is the current value at the time you run the Activity option, not a cumulative total.
Clients	Number of client sessions running. This field is further divided into the type of user: Local, Remote, or Batch. This is the current value at the time you run the Activity option, not a cumulative total.
APWs	Number of APWs running against your database. This is the current value at the time you run the Activity option, not a cumulative total.

Servers

Displays information about server activity.

PROMON Servers Activity display

The following shows a sample Servers Activity display. A separate page is displayed for each server.

```
09/25/03              Activity: Servers
16:28                 from 09/25/03 13:56 to 09/25/03 13:57 (19 sec)
                         Total       Per Min      Per Sec     Per Tx
Messages received          204           131         2.19       0.00
Messages sent              152            98          1.6       0.00
Bytes received           13252          8549       142.49       0.00
Bytes sent               40683         26247       437.45       0.00
Records received             0             0         0.00       0.00
Records sent               122            79         1.31       0.00
Queries received            54            35         0.58       0.00
Time slices                 51            32         0.54       0.00

Activity for Server 1
```

The display contains the following fields:

- **Messages received** — The number of network packets received by the server.
- **Messages sent** — The number of network packets sent by the server.
- **Bytes received** — The number of bytes received by the server.
- **Bytes sent** — The number of bytes sent by the server.
- **Records received** — The number of transferred records received.
- **Records sent** — The number of transferred records sent.
- **Queries received** — The number of query requests received.
- **Time slices** — The number of query time slice switches. A time slice switch occurs when the server moves to processing the next client without completing the query.

Buffer Cache

Displays activity information about the database buffer cache (also called the buffer pool). There are three sets of information: the overall buffer pool, the primary buffer pool, and the Alternate Buffer Pool.

PROMON Buffer Cache Activity display

The following shows a sample Buffer Cache Activity display.

```
06/30/09        Activity: Buffer Cache
11:35:03        06/30/09 11:23 to 06/30/09 11:33 (9 min 23 sec)

                           Total       Per Min   Per Sec    Per Tx
Database Buffer Pool
Logical reads              28841          3074     51.23      7.14
Logical writes             32703          3485     58.09      8.10
O/S reads                    358            38      0.64      0.09
O/S writes                   748            80      1.33      0.19
Checkpoints                    8             1      0.01      0.00
Marked to checkpoint         525            56      0.93      0.13
Flushed at checkpoint        463            49      0.82      0.11
Writes deferred            32161          3427     57.12      7.97
LRU skips                      0             0      0.00      0.00
LRU writes                     0             0      0.00      0.00
APW enqueues                   0             0      0.00      0.00
Database buffer pool hit ratio:  99 %

Primary Buffer Pool
Logicalreads               12862          1371     22.85      3.19
Logical writes             18306          1951     32.52      4.53
O/S reads                     10            11      0.18      0.03
O/S writes                   310            33      0.55      0.08
Marked to checkpoint          25             3      0.04      0.01
Flushed at checkpoint         25             3      0.04      0.01
Writes deferred            18279          1948     32.47      4.53
LRU skips                      0             0      0.00      0.00
LRU writes                     0             0      0.00      0.00
APW enqueue                    0             0      0.00      0.00
Primary buffer pool hit ratio:  99 %

Alternate Buffer Pool
Logical reads              15979          1703     28.38      3.96
Logical writes             14397          1534     25.57      3.57
O/S reads                    256            27      0.45      0.06
O/S writes                   438            47      0.78      0.11
Marked to checkpoint         500            53      0.89      0.12
Flushed at checkpoint        438            47      0.78      0.11
Writes deferred            13882          1479     24.66      3.44
LRU2 skips                     0             0      0.00      0.00
LRU2 writes                    0             0      0.00      0.00
APW enqueues                   0             0      0.00      0.00
Alternate buffer pool hit ratio:  99 %
LRU swaps                      0             0      0.00      0.00
LRU2 replacment policy disabled.
```

The display lists the following types of buffer cache requests:

- **Logical Reads** — The number of client requests for database block read operations.
- **Logical Writes** — The number of client requests for database block write operations.
- **O/S reads** — The number of database blocks read from disk.
- **O/S writes** — The number of database block writes to disk.

- **Checkpoints** — The number of checkpoint operations.
- **Marked at checkpoint** — The number of blocks scheduled to be written before the end of a checkpoint.
- **Flushed at checkpoint** — The number of blocks that were not written during the checkpoint and that had to be written all at once at the end of the checkpoint.
- **Writes deferred** — The total number of changes to blocks that occurred before the blocks were written. Each deferred write is potentially an I/O operation saved.
- **LRU skips** — The number of times a buffer on the LRU chain was skipped because it was locked or modified.
- **LRU writes** — The number of blocks written to free a buffer for a read operation.
- **APW Enqueues** — The number of modified buffers placed on the APW queue for writing.
- **Hit Ratio** — The percentage of buffer cache requests that did not require a physical disk I/O operation.

Page Writers

Displays information about asynchronous page writer (APW) activity.

PROMON Page Writers Activity display

The following shows a sample Page Writers Activity display.

```
09/26/03             Activity: Page Writers
16:29                from 09/25/03 13:56 to 09/26/03 11:23 (21 hrs 27 min)
                                 Total   Per Min   Per Sec    Per Tx

Total DB writes                  3       0         0.00       0.00
APW DB writes                    0       0         0.00       0.00
    scan writes                  8       25        0.42       0.00
    APW queue writes             2       6         0.10       0.00
    ckp queue writes             0       0         0.00       0.00
    scan cycles                  0       0         0.00       0.00
    buffers scanned              0       0         0.00       0.00
    bufs checkpointed            173     0         0.11       0.14
Checkpoints                      8211    0         5.22       6.79
Marked at checkpoint             0       0         0.00       0.00
Flushed at checkpoint            0       0         0.00       0.00

Number of APWs:                   1
```

The display lists the following types of APW operations:

- **Total DB writes** — The total number of database write operations performed by all processes.
- **APW DB writes** — The number of database write operations performed by the APW. This is a subset of the total number of DB writes:
  - **Scan writes** — The number of buffers written during the scan cycle.
  - **APW queue writes** — The number of buffers written to clear the APW queue.
  - **Ckp queue writes** — The number of buffers written from the checkpoint queue.
  - **Scan cycles** — The number of scan cycles. During a scan cycle, the APW scans a portion of the buffer pool to look for modified buffers.
  - **Buffers scanned** — The number of buffers scanned during each cycle.
  - **Bufs checkpointed** — The number of buffers checkpointed.

- **Checkpoints** — The number of checkpoints that have occurred.
- **Marked at checkpoint** — The number of buffers that were scheduled to be written before the end of the checkpoint.
- **Flushed at checkpoint** — The number of blocks that were not written during the checkpoint and had to be written all at once at the end of the checkpoint.
- **Number of APWs** — The number of APWs running.

BI Log

Displays information about BI log activity.

PROMON BI Log Activity display

The following shows a sample BI Log Activity display.

```
09/28/06        Activity: BI Log
15:22:38        09/28/06 10:50 to 09/28/06 15:22 (4 hrs 32 min)

                              Total        Per Min    Per Sec        Per Tx
Total BI writes                1517              6       0.09        758.50
BIW BI writes                   956              4       0.06        478.00
Records written              116270            428       7.13      58135.00
Bytes written               9996664          36768     612.80    4998332.00
Total BI Reads                 386              1       0.02        193.00
Records read                 26923             99       1.65      13461.50
Bytes read                 2330572           8572     142.87    1165286.00
Clusters closed                 19              0       0.00          9.50
Busy buffer waits              588              2       0.04        294.00
Empty buffer waits              29              0       0.00         14.50
Log force waits                  0              0       0.00          0.00
Log force writes                 0              0       0.00          0.00
Partial writes                 222              1       0.01        111.00
Input buffer hits              388              1       0.02        194.00
Output buffer hits             135              0       0.01         67.50
Mod buffer hits                431              2       0.03        215.50
BO buffer hits               53169            196       3.26      26584.50
```

The display lists the following types of BI operations:

- **Total BI writes** — The number of writes to the BI file.
- **BIW BI writes** — The number of writes to the BI file performed by the before-image writer (BIW). For good performance, this number should be high in relation to the total number of BI writes.
- **Records written** — The number of BI records (notes) written to the BI file.
- **Bytes written** — The amount of data written to the BI file, in bytes.
- **Total BI Reads** — The number of BI blocks read from the BI file (to undo transactions).
- **Records read** — The number of BI records (notes) read from the BI file.
- **Bytes read** — The amount of data read from the BI file, in bytes.
- **Clusters closed** — The number of BI clusters filled and closed in preparation for reuse.
- **Busy buffer waits** — The number of times a process had to wait for someone else to finish before being able to write to a BI file.
- **Empty buffer waits** — The number of times a process had to wait because all buffers were full.

- **Log force waits** — The number of waiting-for-commit records to be written to disk.
- **Log force writes** — The number of waiting-for-commit records written to disk.
- **Partial Writes** — The number of writes to the BI file made before the BI buffer is full. This might happen if:
  - The Delayed BI File Write (`-Mf`) parameter timer expired before the buffer was filled.
  - An APW attempts to write a block whose changes are recorded in a BI buffer that has not been written. Because BI notes must be flushed before the AI note is flushed, the AIW writes the data in the BI buffer before the buffer is full so it can do the AI write.
  - An AIW ran ahead of the BIW. Because BI notes must be flushed before the AI notes can be written, the AIW writes the BI buffer before it is full, so it can do the AI write.
- **Input buffer hits** — The number of times a note needed to back out a transaction was found in the Input BI buffer. The Input BI buffer is a shared buffer that contains the BI block needed for a transaction blackout if the note could not be found in the Output BI buffer or the Mod BI buffers. The Input BI buffer is read **back** into memory from the BI log.
- **Output buffer hits** — The number of times a note needed to back out a transaction was found in the Output BI buffer. The Output BI buffer is a shared buffer that contains the BI notes currently being written to disk.
- **Mod buffer hits** — The number of times a note needed to back out a transaction was found in the Mod BI buffers. The Mod BI buffers are shared buffers that are being updated with notes for currently executing transactions. The Mod BI buffers have not been written to disk.
- **BO buffer hits** — The number of times a note needed to back out a transaction was found in a private BO (Back out) buffer. Back out buffers are private buffers allocated, one per user, after transaction blackout processing twice locates the required note in the Input BI buffer. BO buffers reduce contention for the Input BI buffer when more than one user is simultaneously backing out transactions.

AI Log

Displays after-imaging activity.

PROMON AI Log Activity display

The following shows a sample AI Log Activity display.

```
09/13/03             Activity: AI Log
11:36:56             from 09/12/03 13:56 to 09/13/03 11:23 (21 hrs 27 min)

                                    Total   Per Min   Per Sec    Per Tx
Total AI writes                     131     0         0.08       0.10
AIW AI writes                       127     0         0.08       0.10
Records written                     3630    25        2.31       3.00
Bytes written                       129487  0         82.42      107.19
Busy buffer waits                   13      0         0.00       0.01
Buffer not avail                    0       0         0.00       0.00
Partial Writes                      4       0         0.00       0.00
Log force waits                     0       0         0.00       0.00
```

The display lists the following operations:

- **Total AI writes** — The total number of writes to the AI file.
- **AIW AI writes** — The number of AI writes performed by the after-image writer (AIW). This is a subset of the total AI writes.
- **Records written** — The number of records (notes) written to the AI file.

- **Bytes written** — The amount of AI data written to the AI file, in bytes.
- **Busy buffer waits** — The number of times a process had to wait because a buffer was held.
- **Buffer not avail** — The total number of times a process had to wait because a buffer was not available.
- **Partial writes** — The number of writes to the AI file made before the AI buffer is full. This might happen if:
  - The Delayed BI File Write (`-Mf`) parameter timer expired before the buffer was filled
  - An APW attempts to write a block whose changes are recorded in an AI buffer that has not been written
- **Log force waits** — The number of waiting-for-commit records to be written to disk.

Lock Table

Displays information about Lock Table Activity. The database engine stores record locks in the lock table. If a user tries to acquire a lock on a record, and the lock table overflows, the server aborts.

PROMON Lock Table Activity display

The following figure shows a sample Lock Table Activity display.

```
04/29/04        Activity: Lock Table
15:11:21        04/29/04 15:05 to 04/29/04 15:10 (5 min 14 sec)
                              Total   Per Min    Per Sec    Per Tx

Requests:
    Share                      3715       710      11.83     61.92
    Intent Share                  0         0       0.00      0.00
    Exclusive                     8         2       0.03      0.13
    Intent Exclusive              0         0       0.00      0.00
    Share Intent Excl             0         0       0.00      0.00
    Upgrade                       0         0       0.00      0.00
    Record Get Lock            3505       670      11.16     58.42
    Table Lock                   20         4       0.06      0.33
    Record Lock                 198        38       0.63      3.30
Grants:
    Share                      3715       710      11.83     61.92
    Intent Share                  0         0       0.00      0.00
    Exclusive                     8         2       0.03      0.13
    Intent Exclusive              0         0       0.00      0.00
    Share Intent Excl             0         0       0.00      0.00
    Upgrade                       0         0       0.00      0.00
    Record Get Lock            3505       670      11.16     58.42
    Table Lock                   20         4       0.06      0.33
    Record Lock                 198        38       0.63      3.30
Waits:
    Share                         0         0       0.00      0.00
    Intent Share                  0         0       0.00      0.00
    Exclusive                     0         0       0.00      0.00
    Intent Exclusive              0         0       0.00      0.00
    Share Intent Excl             0         0       0.00      0.00
    Upgrade                       0         0       0.00      0.00
    Record Get Lock               0         0       0.00      0.00
    Table Lock                    0         0       0.00      0.00
    Record Lock                   0         0       0.00      0.00

Requests Cancelled                0         0       0.00      0.00
Downgrades                        0         0       0.00      0.00
Shared Find                       0         0       0.00      0.00
Exclusive Find                    0         0       0.00      0.00
```

The display lists the following operations:

- **Requests**
  - **Share** — The number of user requests for a share lock
  - **Intent Share** — The number of user requests for an intent share lock
  - **Exclusive** — The number of user requests for an exclusive lock
  - **Intent Exclusive** — The number of user requests for an intent exclusive lock
  - **Share Intent Exclusive** — The number of user requests for a shared lock with intent exclusive
  - **Upgrade** — The number of user requests to upgrade a lock from shared lock to exclusive lock
  - **Record Get Lock** — The number of user requests for a record get lock
  - **Table Lock** — The number of user requests for a table lock
  - **Record Lock** — The number of user requests for a record lock
- **Grants**
  - **Share** — The number of granted share lock requests
  - **Intent Share** — The number of granted intent share lock requests
  - **Exclusive** — The number of granted exclusive lock requests
  - **Intent Exclusive** — The number of granted intent exclusive lock requests
  - **Share Intent Exclusive** — The number of granted share lock with intent exclusive requests
  - **Upgrade** — The number of granted requests for lock upgrades
  - **Record Get Lock** — The number of granted requests for record get locks
  - **Table Lock** — The number of granted requests for table locks
  - **Record Lock** — The number of granted requests for record locks
- **Waits**
  - **Share** — The number of times users waited for a share lock
  - **Intent Share** — The number of times users waited for an intent share lock
  - **Exclusive** — The number of times users waited for an exclusive lock
  - **Intent Exclusive** — The number of times users waited for an intent exclusive lock
  - **Share Intent Exclusive** — The number of times users waited for a share lock with exclusive intent
  - **Upgrade** — The number of times users waited for a lock upgrade
  - **Record Get Lock** — The number of times users waited for a record get lock
  - **Table Lock** — The number of times users waited for a table lock
  - **Record Lock** — The number of times users waited a record lock
- **Requests cancelled** — The number of lock requests that were cancelled
- **Downgrades** — The number of locks downgraded

I/O Operations By Type

Displays information about I/O activity, organized by type.

PROMON I/O Operations by Type Activity display

The following shows a sample I/O Operations by Type Activity display.

```
09/25/03             Activity: I/O Operations by Type
16:31               from 09/25/03 15:31 to 09/25/03 16:04
                                  Total    Per Min    Per Sec     Per Tx

Database reads                    57       0          0.03        0.04
  Index blocks                    13       0          0.00        0.01
  Data blocks                     44       0          0.02        0.03
BI reads                          13       0          0.00        0.00
AI Reads                          0        0          0.00        0.05
Total reads                       70       0          0.04        0.05

Database writes                   3        0          0.00        0.00
  Index blocks                    0        0          0.00        0.00
  Data blocks                     3        0          0.08        0.00
BI writes                         131      0          0.00        0.10
AI writes                         0        0          0.00        0.00
Total writes                      134      0          0.08        0.11
```

The display lists the following types of I/O operations:

- **Database reads** — Total number of reads from the database:
  - **Index blocks** — The number of reads of index blocks
  - **Data blocks** — The number of reads of data blocks
- **BI reads** — The number of blocks read from BI files
- **AI reads** — The number of blocks read from AI files
- **Total reads** — Total number of database, BI, and AI read operations
- **Database writes** — Total number of writes to the database:
  - **Index blocks** — The number of writes of index blocks
  - **Data blocks** — The number of writes of data blocks
- **BI writes** — The number of BI write operations
- **AI writes** — The number of AI write operations
- **Total writes** — Total number of database, BI, and AI write operations

I/O Operations by File

Displays information about I/O activity, organized by file.

PROMON I/O Operations by File Activity display

The following shows a sample I/O Operations by File Activity display.

```
10/18/06        Activity: I/O Operations by File
12:30:13        10/18/06 12:14 to 10/18/06 12:15 (11 sec)

                                 Total        Per Min     Per Sec     Per Tx
/scratch/shannon/101B/sep25.db
```

```
Reads                                3          16          0.27          0.00
Writes                               0           0          0.00          0.00
Extends                              0           0          0.00          0.00

                                  Total        Per Min     Per Sec     Per Tx
/scratch/shannon/101B/sep25.b1
Reads                               15          82          1.36          0.00
Writes                               1           5          0.09          0.00
Extends                              0           0          0.00          0.00

                                  Total        Per Min     Per Sec     Per Tx
/scratch/shannon/101B/sep25.d1
Reads                               25         136          2.27          0.00
Writes                               3          16          0.27          0.00
Extends                              0           0          0.00          0.00

                                  Total        Per Min     Per Sec     Per Tx
/scratch/shannon/101B/sep25_7.d1
Reads                                1           5          0.09          0.00
Writes                               0           0          0.00          0.00
Extends                              0           0          0.00          0.00

Enter <return> for more, A, L, R, S, U, Z, P, T, or X (? for help):
```

For each file, the display lists the following types of I/O operations:

- **Reads** — The number of read operations performed on the file
- **Writes** — The number of write operations performed on the file
- **Extends** — The number of extend operations performed on the file

Space Allocation

Displays information about Space Allocation.

PROMON Space Allocation Activity display

The following shows a sample Space Allocation Activity display.

```
09/26/03         Activity: Space Allocation
14:52:41         from 09/25/03 13:56 to 09/26/03 11:23 (for 21 hrs 27 min)

                              Total      Per Min    Per Sec    Per Tx
Database extends              57         0          0.03       0.04
Take free block               13         0          0.00       0.01
Return free block             44         0          0.02       0.03
Alloc rm space                13         0          0.00       0.00
Alloc from rm                 0          0          0.00       0.05
Alloc from free               70         0          0.04       0.05
Bytes allocated               3          0          0.00       0.00
rm blocks examined            0          0          0.00       0.00
Remove from rm                3          0          0.00       0.00
Add to rm, front              131        0          0.08       0.10
Add to rm, back               0          0          0.00       0.00
Move rm front to back         134        0          0.08       0.11
Removed locked rm entry       0          0          0.00       0.00
```

The display lists the following operations:

- **Database extends** — The number of times the database was extended

- **Take free block** — The number of times a block was used from the free chain
- **Return free block** — The number of times a block was returned to the free chain
- **Alloc rm space** — The number of times space was allocated for a record or record fragment
- **Alloc from rm** — The number of times space was allocated from the rm chain
- **Alloc from free** — The number of times space was allocated from the free chain
- **Bytes allocated** — The number of bytes allocated for record fragments
- **rm blocks examined** — The number of blocks examined in the rm chain while looking for space for a record fragment
- **Remove from rm** — The number of blocks removed from the rm chain
- **Add to rm, front** — The number of blocks added to the front of the rm chain
- **Add to rm, back** — The number of blocks added to the back of the rm chain
- **Move rm front to back** — The number of blocks moved from the front to the back of the rm chain
- **Remove locked rm entry** — The number of rm chain entries that were locked

Index

Displays information about index activity.

PROMON Index Activity display

The following shows a sample Index Activity display.

```
09/14/03         Activity: Index
17:09:35         from 09/14/03 17:01 to 09/14/03 17:07 (6 min)

                                Total      Per Min    Per Sec    Per Tx
Find index entry                    0            0       0.00      0.00
Create index entry                312            1       0.02    104.00
Delete index entry                  8            0       0.00      2.67
Remove locked entry                 0            0       0.00      0.00
Split block                         0            0       0.00      0.00
Free block                          0            0       0.00      0.00
```

The display shows the following operations:

- **Find index entry** — The number of times an index entry was looked up
- **Create index entry** — The number of new index entries generated
- **Delete index entry** — The number of index entries deleted
- **Remove locked entry** — The number of old locks released at transaction end
- **Split block** — The number of block splits from adding indexes
- **Free block** — The number of index blocks freed during deletes

Record

Displays information about record activity.

PROMON Record Activity display

The following shows a sample Record Activity display.

```
09/14/03         Activity: Record
17:04:35         from 09/14/03 9:24 to 09/14/03 14:38 (5 hrs 14 min)

                                Total    Per Min       Per Sec    Per Tx

Read record                      2374          0          0.13    791.33
Update record                      18          0          0.00      6.00
Create record                      10          0          0.00      3.33
Delete record                       3          0          0.00      1.00
Fragments read                   2610          8          0.14    870.00
Fragments created                  24          0          0.00      8.00
Fragments deleted                  17          0          0.00      5.67
Fragments updated                   4          0          0.00      1.33
Bytes read                     409795       1307         21.7   136598.33
Bytes created                    1813          6          0.10    604.33
Bytes deleted                    1310          4          0.07    436.67
Bytes updated                    1042          3          0.06    347.33
```

The display shows the following operations:

- **Read record** — The number of records read
- **Update record** — The number of records modified
- **Create record** — The number of records created
- **Delete record** — The number of records deleted
- **Fragments read** — The number of fragments read
- **Fragments created** — The number of fragments created
- **Fragments deleted** — The number of fragments deleted
- **Fragments updated** — The number of fragments updated
- **Bytes read** — The number of bytes read
- **Bytes created** — The number of record bytes created
- **Bytes deleted** — The number of record bytes deleted
- **Bytes updated** — The number of record bytes updated

Other

Displays information about miscellaneous activity.

PROMON Other Activity display

The following shows a sample Other Activity display.

```
09/14/03          Activity: Other
17:04:35          from 09/14/03 17:01 to 09/14/03 17:01 (10 sec)

                                Total       Per Min     Per Sec     Per Tx

Commit                          3           0           0.00        1.00
Undo                            0           0           0.00        0.00
Wait on semaphore               0           0           0.00        0.00
Non-blocking waits
Semaphore latch waits
Flush master block              3           18          0.30        1.00
```

The display shows the following operations:

- **Commit** — The number of transactions committed
- **Undo** — The number of transactions rolled back
- **Wait on semaphore** — The number of times a process had to wait for a resource
- **Non-blocking waits —** The number of non-blocking waits for remote clients
- **Semaphore latch waits —** The number of semaphore latch waits
- **Flush master block** — The number of times the database master block was written to disk

R&D Other Displays

The **R&D Other Displays menu**provides a selection of miscellaneous options.

- Performance Indicators on page 607
- I/O Operations by Process on page 609
- Lock Requests by User on page 609
- Checkpoints on page 610
- I/O Operations by User by Table on page 612
- I/O Operations by User by Index on page 612
- Total Locks per User on page 613

Performance Indicators

Displays activity statistics related to performance.

PROMON Performance Indicators display

The following shows a sample Performance Indicators display.

```
09/19/03         Activity: Performance Indicators
17:04:35         from 09/15/03 10:01 to 09/19/03 13:26 (99 hrs 24 min)

                                Total      Per Min    Per Sec    Per Tx
Commits                         0          0          0.00       0.00
Undos                           0          0          0.00       0.00
Index Operations                0          0          0.00       0.00
Record Operations               3          18         0.30       0.00
Total o/s i/o                   28         0          0.00       0.00
Total o/s reads                 23         0          0.00       0.00
Total o/s writes                5          0          0.00       0.00
Background o/s writes           5          0          0.00       0.00
Partial log writes              1          0          0.00       0.00
Database extends                0          0          0.00       0.00
Total waits                     0          0          0.00       0.00
Lock waits                      0          0          0.00       0.00
Resource waits                  0          0          0.00       0.00
Latch timeouts                  0          0          0.00       0.00
buffer pool hit rate: 68%
```

The Performance Indicators display shows the total number of each type of operation for the sample period (**Total**), the number of operations per minute (**Per Min**) the number of operations per second (**Per Sec**), and the number of operations per transaction (**Per Tx**). The display shows the following operations:

- **Commits** — The number of committed transactions.
- **Undos** — The number of transactions that were rolled back.
- **Index Operations** — The total of all operations performed on indexes (for example, index additions and deletions).
- **Record Operations** — The total of all operations on records (for example, record additions and deletions).
- **Total o/s i/o** — The total number of read and write operations performed.
- **Total o/s reads** — The total number of read operations performed.
- **Total o/s writes** — The total number of write operations performed.
- **Background o/s writes** — The total number of writes performed by background writers (APWs, BIW, and AIW). See Managing Performance on page 275 for more information on background writers.
- **Partial log writes** — Writes to the BI file made before the BI buffer is full. This might happen if:
  - The Delayed BI File Write (`-Mf`) parameter timer expired before the buffer was filled.
  - An APW attempts to write a block whose changes are recorded in a BI buffer that has not been written. Because BI notes must be flushed before the AI note is flushed, the AIW writes the data in the BI buffer before the buffer is full so it can do the AI write.
  - An AIW ran ahead of the BIW. Because BI notes must be flushed before the AI notes can be written, the AIW writes the BI buffer before it is full, so it can do the AI write.
- **Database extends** — The total number of times the database was made larger by allocating operating system blocks.
- **Total waits** — The total number of waits.
- **Lock waits** — The number of times the database engine waited for a lock to be released.

- **Resource waits** — The number of times the database engine waited for a resource, such as a row lock, a buffered lock in shared memory, or a transaction end lock, to become available.
- **Latch timeouts** — The number of times a spin lock expired, causing the process to nap, when attempting to acquire a latch.
- **Buffer pool hit rate** — The percentage of times that the database engine found a record in the buffer pool and did not have to read the record from disk.

I/O Operations by Process

Displays information about database I/O operations, organized by process.

PROMON I/O Operations by Process display

The following shows a sample I/O Operations by Process display.

```
09/25/11           I/O Operations by Process
16:19

                              ---------Database -------     ---- BI -----     ---- AI -----
Usr:Ten   Name     Domain    Access     Read     Write     Read    Write     Read    Write

  0:0   shannon          0       815       77         81       25        1         0        0
  5:0   shannon          0        16       77         81       25        1         0        0
  6:ST  ST_bus           1      1746       29          1        0        1         0        0
```

This display shows the following information:

- **Usr:Ten** — The user number and tenant of the process
- **Name** — The user name of the process
- **Domain** — The domain of the user
- **Database Access** — The number of database access operations performed by the process
- **Database Read** — The number of database read operations performed by the process
- **Database Write** — The number of database write operations performed by the process
- **BI Read** — The number of BI read operations performed by the process
- **BI Write** — The number of BI write operations performed by the process
- **AI Read** — The number of AI read operations performed by the process
- **AI Write** — The number of AI write operations performed by the process

Lock Requests by User

Displays information about lock requests, organized by user.

PROMON Lock Requests by User display

The following shows a sample Lock Requests by User display.

```
09/25/03        Lock Requests by User
16:19

Usr:Ten   User    Domain  --- Record ---   ---- Trans ---    ---- Schema ---
          Name            Locks    Waits   Locks    Waits    Locks     Waits
  0:0    SYSTEM        0      0        0       0        0        0         0
  5:0   shannon        0      0        0       0        0        0         0
  6:ST   st_bus        1      0        0       0        0        0         0
```

This display shows the following information:

- **Usr:Ten** — The user number and tenant
- **User Name** — The user name
- **Domain** — The domain of the user
- **Record Locks** — The number of records locks acquired by the user
- **Record Waits** — The number of times the user had to wait for a record lock
- **Trans Locks** — The number of times a deleted record is protected until transaction end
- **Trans Waits** — The number of times a user had to wait for a record marked for deletion by an active transaction
- **Schema Locks** — The number of schema locks acquired by the user
- **Schema Waits** — The number of times the user had to wait for a schema lock

Checkpoints

Displays information about checkpoints.

PROMON Checkpoints display

The following shows a sample Checkpoints display.

```
03/24/15        Checkpoints
17:50:12

Ckpt                          --------- Database Writes ----------  ------ Performance Timings
 ---------
 No. Time      Len Freq Dirty CPT Q Scan APW Q DB Writes Bi Writes Duration Sync Time DB Write
 Bi Write NumChkpt
  56 17:50:07   49    0    24     0    0     0         0        19     0.02      0.00     0.00
     0.01       13

  55 17:50:07    0    0    27     0    0     0        13        19     0.02      0.00     0.00
     0.01       14
  54 17:50:03    4    4    21     0    0     0        14        19     0.02      0.00     0.00
     0.02        7
  53 17:50:03    0    0    22     0    0     0         7        19     0.02      0.01     0.00
     0.02       15
  52 17:50:02    1    1    32     0    0     0        15        19     0.03      0.00     0.00
     0.02       17
  51 17:49:14   48   48    30     0    0     0        17         9     0.01      0.00     0.00
     0.01       13
  50 17:49:07    7    7    17     0    0     0        13        19     0.02      0.00     0.00
     0.01        4
  49 17:49:06    1    1    24     0    0     0         4        19     0.02      0.00     0.00
     0.01       20

Enter <return>, R, U, P, T, or X (? for help):
```

This display shows the following information:

- **Ckpt. No.** — The sequential number assigned to the checkpoint
- **Time** — The time the checkpoint began
- **Len** — The length of time required to complete the checkpoint. The checkpoint is not considered completed until all dirty buffers on the checkpoint queue have been written out. This is either done over time by the APWs between checkpoints or is forced at the next checkpoint (**DB Writes**).
- **Freq** — The length of time between checkpoint starts
- **Dirty** — The number of modified buffers scheduled to be written
- **Database Writes**
  - **CPT Q** — The number of buffers written from the checkpoint queue by the APWs
  - **Scan** — The number of buffers written by the APWs during the scan cycle
  - **APW Q** — The number of buffers written by the APW queue and replaced on the least recently used (LRU) chain by APWs
  - **DB Writes** — The total number of database buffers written all at once at the end of the checkpoint
  - **Bi Writes** — The total number of BI buffers written from the BI buffer pool all at once at the end of the checkpoint
- **Performance Timings**
  - **Duration** — The time to do the housekeeping of the checkpoint. This includes the time to write the active recovery log buffers to disk, write any outstanding dirty buffers marked at the previous checkpoint, scan the database buffer pools for dirty buffers marking them as needing to be written to disk prior to the start

of the next check point (putting them on the checkpoint queue) and to perform the file system synchronization (**Sync Time**).

- **Sync Time** — The time to perform the file system synchronization of the checkpoint (`fdatasync()` on UNIX, `FlushFileBuffers()` in Windows)
- **DB Write** — The time in seconds to scan and write the buffers from the database buffer pool
- **Bi Write** — The time in seconds to write the buffers from the -bibufs buffer pool
- **NumChkpt** — The number of buffers checkpointed (written from the database buffer pool)

I/O Operations by User by Table

Displays I/O operations by user and by table.

PROMON I/O Operations by User by Table display

The following shows a sample I/O Operations by User by Table display.

```
09/21/06        Activity: I/O Operations by User by Table
13:41:38        09/21/06 09:36 to 09/21/06 09:37 (39 sec)

 Usr:Ten    Table        Reads             Updates          Creates      Deletes
   0:0          1            0                   0                0            0
   0:0          2            0                   0                0            0
   0:0          3            0                   0                0            0
   0:0          4            0                   0                0            0
   0:0          5            0                   0                0            0
   0:0          6            0                   0                0            0
   5:0          1            0                   0                0            0
   5:0          2            0                   0                0            0
   5:0          3            0                   0                0            0
   5:0          4            0                   0                0            0
   5:0          5            0                   0                0            0
   5:0          6            0                   0                0            0

Enter <return> for more, A, L, R, S, U, Z, P, T, or X (? for help):
```

This display shows the following information:

- **User:Ten** — The user number and tenant
- **Table** — The table number
- **Reads** — The number of reads the user has performed on the table
- **Updates** — The number of updates the user has performed on the table
- **Creates** — The number of creates the user has performed on the table
- **Deletes** — The number of deletes the user has performed on the table

By default, the display will show the user activity for up to 100 tables. The monitored tables are selected by the startup parameters `-basetable` and `-tablerange`. See Database Startup Parameters on page 473 for more information on these parameters.

I/O Operations by User by Index

Displays information regarding I/O operations by user and by index.

PROMON I/O Operations by User by Index display

The following shows a sample I/O Operations by User by Index display.

```
Usr:Ten    Index      Reads      Creates      Deletes      Splits      Block Dels
  0:0          1          0            0            0           0               0
  0:0          2          0            0            0           0               0
  0:0          3          0            0            0           0               0
  0:0          4          0            0            0           0               0
  0:0          5          0            0            0           0               0
  0:0          6          0            0            0           0               0
  0:0          7          0            0            0           0               0
  0:0          8          0            0            0           0               0
  0:0          9          0            0            0           0               0
  5:0          1          0            0            0           0               0
  5:0          2          0            0            0           0               0
  5:0          3          0            0            0           0               0
  5:0          4          0            0            0           0               0
  5:0          5          0            0            0           0               0
  5:0          6          0            0            0           0               0
  5:0          7          0            0            0           0               0
  5:0          8          0            0            0           0               0
  5:0          9          0            0            0           0               0
  6:ST         1          0            0            0           0               0
  6:ST         2          0            0            0           0               0
  6:ST         3          0            0            0           0               0
  6:ST         4          0            0            0           0               0
  6:ST         5          0            0            0           0               0
  6:ST         6          0            0            0           0               0
  6:ST         7          0            0            0           0               0
  6:ST         8          0            0            0           0               0
  6:ST         9          0            0            0           0               0

Enter <return> for more, A, L, R, S, U, Z, P, T, or X (? for help):
```

This display shows the following information:

- **User:Ten** — The user number and tenant
- **Index** — The index number
- **Reads** — The number of times read access has occurred to the index by the user
- **Creates** — The number of times create access has occurred to the index by the user
- **Deletes** — The number of times delete access has occurred to the index by the user
- **Splits** — The number of split operations that have occurred to the index by the user
- **Block dels** — The umber of block deletes that have occurred to the index by the user

By default, the display will show the user activity for up to 100 indexes. The monitored indexes are selected by the startup parameters `-baseindex` and `-indexrange`. See Database Startup Parameters on page 473 for more information on these parameters.

Total Locks per User

Displays, for each user, the total number of locks and a count of the various types of locks.

PROMON Total Locks per User display

The following figure shows a sample Total Locks per User display.

```
08/05/15        Status: Total Locks per User by user number
16:24:21
Usr:Ten Name  Domain  Type       PID TTY    Total   Table Partition  Record  Purge RecGet
SHR  EXCL   IS   IX  SIX

  5:0   ees   -3      SELF/ABL  5688 CON:       4       1         1       2      0      0
 3     1    0    2    0
  6:0   abs   -3      SELF/ABL  1696 CON:       2       0         0       2      0      0
 1     1    0    0    0
```

R&D Administrative Functions

When you enter `4` to select the **Administration Functions** in the R&D advanced options, the following menu items appear. You can view and change the settings using these menu items.

- Check Active Transaction Status on page 614
- Check Two-Phase Transactions on page 615
- Resolve Limbo Transactions on page 615
- Adjust Latch Options on page 616
- Adjust Page Writer Options on page 616
- Restricted Options on page 617
- Server Options on page 617
- Enable/disable block level consistency check on page 618
- Shutdown Database on page 619
- Disconnect a tenant's users on page 619
- Adjust Governor options on page 620
- Toggle redundantlockdiag setting (for use when directed by Progress Technical Support only)
- Caches on page 620
- Diagnostic Data Collection on page 621

Check Active Transaction Status

Displays information about active transactions.

PROMON Check Active Transaction Status display

The following shows a sample display.

```
12/18/11        OpenEdge Release 11 Monitor (R&D)
16:19:34          Check Active Transaction Status

Usr:Ten  Name    Domain   Txt start time   Trans id    Trans State   Crd Nam    Crd Tx Id
  8:0    rjw           1  09/25/11 16:12        178    Active        -          0
```

The display shows the following information:

- **Usr:Ten** — The user number a tenant of the process
- **Name** — The user name of the process
- **Domain** — The Domain of the user
- **Txt start time** — The transaction's start time
- **Trans id** — The transaction's ID
- **Trans State** — The transactions state. Table 83: Transaction states on page 555 lists possible transaction states
- **Crd Nam** — Name of the coordinator database
- **Crd Tx Id** — Transaction ID in the coordinator database

Check Two-Phase Transactions

Displays information about Two-Phase Commit.

Resolve Limbo Transactions

Lets you resolve a limbo transaction by either committing or aborting the transaction.

PROMON Resolve Limbo Transactions menu

When you choose this option, PROMON displays the following menu.

```
1 Abort a Limbo Transaction
2 Commit a Limbo Transaction
Q Quit

Enter choice>
```

Choose one of the following options:

- **Abort a Limbo Transaction** — When you choose **Abort a Limbo Transaction**, PROMON prompts you to enter the user number of the transaction you want to abort. PROMON displays the user's number in the **User** column of the **Transaction Control option** screen. Type the user number, then press **RETURN**.
- **Commit a Limbo Transaction** — When you choose **Commit a Limbo Transaction**, PROMON prompts you to enter the user number of the transaction you want to commit. (PROMON displays the user number in the **Usr** column of the **Transaction Control option** screen.) Type the user number, then press **RETURN**. PROMON displays a message similar to the following:

  ```
  User 1: commit transaction and disconnect.
  ```

  Note: To commit transactions on a database that is shut down, you must use the 2PHASE RECOVER qualifier of PROUTIL.

Adjust Latch Options

Adjusts the Spin Lock mechanism used to control access to shared system resources.

PROMON Adjust Latch Options menu

The following shows a sample Adjust Latch Options menu.

```
01/25/02        OpenEdge Monitor (R&D)
16:34:27        Adjust Latch Options

                1. Spins before timeout         144000
                2. Record Free Chain Search Depth Factor: 5
                3. Enable LRU2 alternate buffer pool replacement policy
                4. Adjust LRU force skips:           0
                5. Adjust LRU2 force skips:          0
```

This menu contains the following:

- **Spins before timeout** — Specifies a value to dynamically tune the Spin Lock mechanism. This mechanism can also be tuned using the Spin Lock Tries (`-spin`) parameter. For more information on spin locks, see Spin locks on page 306.
- **Record Free Chain Search Depth Factor** — Displays the current setting for the searching the record chain for free space.
- **Enable(Disable) LRU2 alternate buffer pool replacement policy** — Allows you to toggle the LRU replacement policy for the alternate buffer pool
- **Adjust LRU force skips** — Specifies the number of times a buffer in the buffer pool is accessed before it is placed on the Most Recently Used (MRU) end of the LRU chain.
- **Adjust LRU2 force skips** — Specifies the number of times a buffer in the alternate buffer pool is accessed before it is placed on the Most Recently Used (MRU) end of the LRU chain.

Adjust Page Writer Options

Lets you manually override APW behavior. (This is rarely necessary. Do not change these options unless directed to do so by Technical Support.)

PROMON Adjust Page Writer Options menu

The following shows a sample Adjust Page Writer Options menu.

```
01/25/02        OpenEdge Monitor (R&D)
16:19:34        Adjust Page Writer Options

               1. APW queue check time:        100 milliseconds
               2. APW buffers scan time:       1 seconds
               3. APW buffers per scan:        1
               4. APW writes per scan:         25
```

Choose one of the following options:

- **APW queue check time** — Allows you to dynamically adjust the value of the Page Writer Queue Delay (`-pwqdelay`) parameter.

Note: Though you can set the value of `-pwqdelay`, it is a self-tuning parameter. When the database encounters heavy loads, `-pwqdelay` decreases its value so that the APW writes more often. When the demand on the database lessens, `-pwqdelay` increases its value so that the APW writes less often. The default value, in milliseconds, for `-pwqdelay` is 100.

- **APW buffers scan time** — Allows you to dynamically adjust the value of the Page Writer Scan Delay (`-pwsdelay`) parameter. The default value, in seconds, for `-pwsdelay` is 1.
- **APW buffers per scan** — Enables you to dynamically adjust the value of the Page Writer Scan (`-pwscan`) parameter.
- **APW writes per scan** — Enables you to dynamically adjust the value of the Page Writer Maximum Buffers (`-pwwmax`) parameter.

Restricted Options

This option can only be used under the supervision of Progress Technical Support. Selecting this option will prompt you for a **restricted option key** which Technical Support will provide if necessary.

Server Options

This option allows you to change the server options or terminate a server.

PROMON Server Options display

The following shows the Server Options display:

```
11/05/18        OpenEdge Release 12 Monitor (R&D)
16:36:49        Server Options

                 1. Server network message wait time:            2 seconds
                 2. Delay first prefetch message:                Disabled
                 3. Prefetch message fill percentage:            0 %
                 4. Minimum records in prefetch message:         16
                 5. Suspension queue poll priority:              0
                 7. Terminate a server
                 8. User notify time:                            0 seconds
                 9. Limit .lg message payload:                   Disabled
                10. SQL Autonomous Statistics:                  ON

Enter a number, P, T, or X (? for help):
```

Choose one of the following options:

- **Server network message wait time** — Enables you to dynamically adjust the value of the Server network message wait time (`-Nmsgwait`) parameter.
- **Delay first prefetch message** — Enables you to dynamically adjust the value of the Delay first prefetch message (`-prefetchDelay`) parameter.
- **Prefetch message fill percentage** — Enables you to dynamically adjust the value of the Prefetch message fill percentage (`-prefetchFactor`) parameter.
- **Suspension queue poll priority** — Enables you to dynamically adjust the value of the Suspension queue poll priority (`-prefetchNumRecs`) parameter.

- **Terminate a Server** — Enables you to select a server for termination. The following shows the Terminate a Server display.

```
12/18/11        OpenEdge Release 11 Monitor (R&D)
15:38:45        Terminate a Server

Sv                                      Pend.   Cur.   Max.    Port
No    Pid  Type       Protocol Logins  Users  Users  Users    Num
  0  19952 Login                   0      0      0      4      0
  1  17872 Login      TCP          0      0      0      5   2054
  2      0 Inactive               0      0      0      0      0
  3      0 Inactive               0      0      0      0      0
  4      0 Inactive               0      0      0      0      0

Enter server number, <return> (to continue), or P (previous menu):
```

- **Limit .lg message payload** — Enables you to dynamically toggle the value of the Limit log file payload (`-limitLgPayload`) parameter. By default, the parameter is disabled, and the payload information (user name, tty name, IP address, and connection hostname) is written to the log file. Toggle the setting by selecting `9`.
- **SQL Autonomous Statistics** - The AUS server status is `ON`. Toggle the setting by selecting `10`.

```
11/05/18        OpenEdge Release 12 Monitor (R&D)
14:31:48        Server Options

                 1. Server network message wait time:            2 seconds
                 2. Delay first prefetch message:                Disabled
                 3. Prefetch message fill percentage:            0 %
                 4. Minimum records in prefetch message:         16
                 5. Suspension queue poll priority:              0
                 7. Terminate a server
                 8. User notify time:                            0 seconds
                 9. Limit .lg message payload:                   Disabled
                10. SQL Autonomous Statistics:                  ON

Enter a number, P, T, or X (? for help):
```

Enable/disable block level consistency check

This setting enables memory overwrite checking at runtime and during roll forward. This check focuses on potential memory overwrites in the buffer pool. When this setting is enabled, the consistency checks will be applied to all index blocks and record blocks.

Note: Memory overwrite checking is performed on record and index write operations only.

When using this option, consider the following:

- Index and record block operations are checked prior to before-image note writing and the subsequent database block write. The length of the intended operation is checked to prevent operations that exceed the database block size.
- Index block splits are checked during the strategic points in an index block split operation.
- Index insert operations are checked during the strategic points of the insertion of a key element.

PROMON enable/disable block level consistency check display

The following displays available settings.

```
12/18/11        OpenEdge Release 11 Monitor (R&D)
10:19:51        Eanble/Disable block level consistency check

                This option is used to enable/disable block consistency check.

                Current consistency check status:

                1. -MemCheck:   disabled
                2. -DbCheck:    disabled
                3. -AreaCheck:  disabled
                4. -IndexCheck: disabled
                5. -TableCheck: disabled

Enter the option to enable/disable a consistency check:
```

Choose one of the following settings:

- **-MemCheck** — Enables memory consistency check.
- **-DbCheck** — Enables database consistency check. When this option is enabled, the consistency check is applied to all index blocks and record blocks (except BLOB blocks) in the entire database.
- **-AreaCheck** — Enables area consistency check. When this option is enabled, the consistency check is applied to all index blocks and record blocks (except BLOB blocks) in the specified area.
- **-IndexCheck** — Enables index consistency check. When this option is enabled, the consistency check is applied to all index blocks of the specified index. This check is for index blocks only.
- **-TableCheck** — Enables table consistency check. When this option is enabled, the consistency check is applied to all record blocks (except BLOB blocks) of the specified table. This check is for record blocks only.

The following precedence rules apply when enabling multiple parameters at the same time:

- Each option can only be enabled once. For example, the following parameters are invalid: `-AreaCheck` *`Customer Area`* `-AreaCheck` *`Order Area.`*
- If the `-DbCheck` option is used with any other options, `-DbCheck` will have precedence and the consistency check will be applied on the whole database.
- `-AreaCheck <` *`area name`*`>`, `-IndexCheck <`*`index name`*`>` and `-TableCheck <`*`table name`*`>` can be used at the same time. The consistency check will be enabled for the superset of these options.

Shutdown Database

This option is identical to other database shutdown interfaces in PROMON. See PROMON Database Status option on page 562 for details.

Disconnect a tenant's users

Displays the currently connected tenants and provides a mechanism for selecting the tenant to disconnect.

PROMON Currently Connected Tenants

The following figure shows a sample display.

```
10/27/11      Currently Connected Tenants
06:29:07

   Sel        Tenant Id        Name           User Count
     1             -1         super_tenant         1

Enter tenant row number or A for all, P, T, or X (? for help):
```

The display shows the following information:

- **Sel** — A row number for selecting the tenant to disconnect users
- **Tenant ID** — The tenant Id
- **Name** — The name of the tenant
- **User Count** — The number of users currently connected for this tenant

Disconnect users by entering a number from the **Sel** column to specify a tenant, or enter **A** to disconnect all tenants.

Adjust Governor options

Displays the current settings of the multi-tenant governors.

PROMON Adjust Governor options

The following shows a sample display.

```
05/24/12        OpenEdge Release 11 Monitor (R&D)
18:34:57        Adjust Governor options

1. -nGovernor:        20 (of 21)
2. -LGovernor:        100% (8192 of 8192)

Enter a number, P, T, or X (? for help):
```

Choose one of the following options:

- **-nGovernor** — Enables you to dynamically adjust the value of the Login Governor (`-nGovernor`) parameter
- **-LGovernor** — Enables you to dynamically adjust the value of the Lock Governor (`-LGovernor`) parameter

Caches

Displays information on shared caches.

PROMON Caches

The following shows a sample display.

```
03/08/17        Status: Caches
14:03:33

 ID        Name      Size(kb)  Entries  Hash      Hits        Misses
--- --------------- -------- -------- ------ ------------ ------------
  1 CDC Primary           100        0    317            0            0
  2 CDC Secondary         100        0    317            0            0

Enter Cache ID to reset total hits and total misses


Enter <return>, R, P, T, or X (? for help):
```

The display shows the following information:

- **ID** — The ID of the cache
- **Name** — The cache name
- **Size (kb)** — The cache size
- **Entries** — The number of entries in the cache
- **Hash** — The size of the hash table for locating values in the cache
- **Hits** — The total number of cache hits since the last reset
- **Misses** — The total number of cache misses since the last reset

To reset the hits and misses count for a cache:

1. Enter the **ID** of the cache at the prompt. You are prompted to confirm your selection.
2. Enter `Y` to reset the hits and misses counts of the selected cache, or enter `N` to continue without resetting the values.

Diagnostic Data Collection

Displays current Diagnostic Data Collection settings and allows you to change the settings.

PROMON Diagnostic Data Collection

The following shows a sample display:

```
11/02/17        OpenEdge Release 12 Monitor (R&D)
12:02:06        Diagnostic Data Collection

        Current Diagnostic Data Collection Settings:

         1. Diagnostic directory:       ""
         2. Diagnostic field separator: ' '
         3. Diagnostic pause time:      0
         4. Diagnostic prefix value:    diagEvent_
         5. Diagnostic report format    csv

         6. Lock Table Overflow:       0: Data collection disabled
         7. BI Threshold:              0: Data collection disabled
         8. System Error:              0: Data collection disabled

         9. Collect enabled diagnostic data now


Enter a number, P, T, or X (? for help):
```

For items **1-5**, change any value by entering the number, and entering a new value as prompted.

For items **6-8**, the following menu prompts you to set a value for the data collection.

```
Enter 0: Disabled
      1: Summary
      2: Summary & Detailed
      3: Summary, Detailed & Protraces
    4-7: Proc invocation in combination w/chosen option 0 thru 3
   8-15: Report all enabled event collectors in combination w/chosen option 0 thru 4
  (-15 to 15):
```

Note that negative values collect the same amount of data, but only in the case of a fatal error that will bring down the database.

For item **9**, the data collection executes immediately, and does not respect the pause time value.

R&D Adjust Monitor Options

Displays parameters controlling the PROMON display and allows you to change them.

PROMON Adjust Monitor Options menu

The following shows the Adjust Monitor Options menu.

```
12/18/11        OpenEdge Release 11 Monitor (R&D)
16:19:34        Adjust Monitor Options

               1. Display page length:           : 24 lines
               2. Clear screen for first page:   : Yes
               3. Monitor sampling interval:     : 10 seconds
               4. Pause between displays:        : 10 seconds
               5. Pause between screens:         : 5 seconds
               6. Number of auto repeats:        : 10
               7. Change working area:           : All areas
               8. Sort user lists                : by user number
               9. Tenant filter for user lists   : None
```

Adjust Monitor Options displays the following information:

- **Display page length** — The number of lines on the terminal screen
- **Clear screen for first page** — If Yes, the system clears the screen before the main menu and before every list
- **Monitor sampling interval** — The interval at which PROMON samples data, in seconds
- **Pause between displays** — The delay between pages of output when in continuous monitoring mode, in seconds (type `u` for the Continue uninterrupted option)
- **Pause between screens** — The delay after the last page of output when in continuous monitoring mode, in seconds
- **Number of auto repeats** — The number of times the display is repeated when auto repeat is selected on the activity displays
- **Change working area** — Displays the area being monitored; by default, PROMON monitors all areas
- **Sort user lists** — Toggles the user list sort between sorting by user number or tenant Id
- **Tenant filter for user lists** — Allows you to filter by one or many tenants using the **Currently Connect Tenants screen**.

Note: To change a monitor option value, enter the option number, and then enter the new value as prompted.

PROMON 2PC Transactions Control option

Displays information about distributed two-phase commit transactions in a database.

Sample output for PROMON 2PC Transaction Control option

The following shows an example of this option's output.

```
Transaction Control:
Usr  Name   Trans   Login   Time   R-comm?   Limbo?   Crd?  Coord  Crd-task
  2  paul   757  07/25/01   10:15    yes      yes      no   sports1   760
      .    .     .      .           .   .         .         .     .      .
      .    .     .      .           .   .         .         .     .      .
      .    .     .      .           .   .         .         .     .      .

RETURN - repeat, U - continue uninterrupted, Q - quit
```

Note: If you run PROMON against a database where no limbo transaction has occurred, PROMON does not display any field information on the **Transaction Control** screen.

`Usr`

The number of the user running the distributed transaction.

`Name`

The name of the user running the distributed transaction.

`Trans`

The transaction number on the machine where you are running PROMON.

`Login`

The date the user running the distributed transaction began the session.

`Time`

The time the user running the distributed transaction began the session.

`R-comm?`

The state of the transaction. If this field is yes, the transaction is in a ready-to-commit state. This means that the first phase of two-phase commit is complete.

`Limbo?`

If a transaction is a limbo transaction. If one or more limbo transactions occur, the **Limbo** field displays a yes for each transaction listed. A yes in this field means that you must resolve transactions.

`Crd?`

If the database you are running PROMON against is the coordinator.

`Coord`

The name of the coordinator database.

`Crd-task`

The transaction number of the coordinator database.

PROMON Resolve 2PC Limbo Transactions option

Lets you resolve a two-phase commit limbo transaction by either committing or aborting the transaction.

Sample output for PROMON Resolve 2PC Limbo Transactions option

When you choose this option, PROMON displays the menu shown in the following figure.

```
1 Abort a Limbo Transaction
2 Commit a Limbo Transaction
Q Quit

Enter choice>
```

`Abort a Limbo Transaction`

When you choose `Abort a Limbo Transaction`, PROMON prompts you to enter the user number of the transaction you want to abort. (PROMON displays the user's number in the **User** column of the **2PC Transaction Control** screen.) Enter the user's number, then press **RETURN**.

`Commit a Limbo Transaction`

When you choose `Commit a Limbo Transaction`, PROMON prompts you to enter the user number of the transaction you want to commit. (PROMON displays the user number in the **Usr** column of the **2PC Transaction Control** screen.) Enter the user number, then press **RETURN**. PROMON displays a message similar to the following:

```
User 1: commit transaction and disconnect.
```

Note: To commit transactions on a database that is shut down, you must use the 2PHASE RECOVER qualifier of PROUTIL.

PROMON 2PC Coordinator Information option

Lets you determine whether a limbo transaction was committed. You need this information before resolving a limbo transaction.

To get this information, run PROMON against the transaction's coordinator database. PROMON displays the name of a transaction's coordinator database in the **Coord** column of the **2PC Transaction Control** screen.

Note: If the coordinator database is shut down and you cannot run PROMON against it, you must use the 2PHASE COMMIT qualifier of PROUTIL to determine whether it committed the transaction.

Sample output for PROMON Coordinator Information option

The following shows an example of this option's output.

```
OpenEdge MONITOR Version 11

    Database: /users/sports1

    Q. QUIT

Enter the transaction number you want to find out if committed:
```

PROMON displays the transaction number in the **Crd-task** column of the **Transaction Control** screen. Enter the transaction number and press **RETURN**. PROMON displays a message that tells you whether the transaction committed.

PROMON Resolve JTA Transactions option

Let you display, commit, or rollback a JTA transaction.

PROMON Resolve JTA Transactions menu

The following displays the **Resolve JTA Transactions** menu.

```
WARNING:  Committing or rolling back a JTA transaction can compromise
WARNING:  referential integrity.  Proceed with EXTREME caution.
     1.  Display all JTA Transactions
     2.  Rollback a JTA Transaction
     3.  Commit a JTA Transaction

     Q.  Return to main menu

     Enter your selection:
```

`Display all JTA Transactions`

When you choose **Display all JTA Transactions**, PROMON displays the following information about all the active JTA transactions: internal transaction id, user id, JTA state, and external JTA transaction id.

`Rollback a JTA Transaction`

When you choose **Rollback a JTA Transaction**, PROMON prompts you for a transaction id to rollback and requires you to confirm your request before rolling back the transaction.

Caution: Rolling back a JTA transaction independent of the external transaction manager, can compromise referential integrity. Choose this option only when your JTA transaction manager has encountered a non-recoverable failure.

`Commit a JTA Transaction`

When you choose **Commit a JTA Transaction**, PROMON prompts you for a transaction id to commit and requires you to confirm your request before committing the transaction.

Caution: Committing a JTA transaction independent of the external transaction manager, can compromise referential integrity. Choose this option only when your JTA transaction manager has encountered a non-recoverable failure.

PROMON Modify Defaults option

Displays terminal behavior parameters and lets you change them when running the monitor. It also displays and lets you change the value of startup parameters used for tuning the asynchronous page writer.

PROMON Modify Defaults menu

The following shows the menu that appears when you choose the **Modify Defaults** option.

```
Enter your selection: m
1. Page size                    : 24
2. Clear screen for first page  : Yes
3. Short pause after each page  : 4
4. Long pause after last page   : 8
5. Monitor Sampling Interval    : 30 sec.
6. APW queue delay              : 500 ms.
7. APW queue start              : 1
8. APW scan delay               : 3 sec.
9. APW scan count               : 9
A. APW write limit              : 5
B. BIW scan delay               : 0
C. Group commit delay           : 10
S. Sort user lists              : by user number
T. Tenant filter for user lists : none
Q. Return to main menu
Enter your selection:
```

The Modify Defaults menu provides the following options:

- **Page size** — The number of lines on the terminal screen.
- **Clear screen for first page** — If Yes, the system clears the screen before the main menu and before every list.
- **Short pause after each page** — The delay in seconds between pages of output when in continuous monitoring mode. (Enter `u` for the Continue Uninterrupted option.)
- **Long pause after last page** — The delay after the last page of output when in continuous monitoring mode.
- **Monitor Sampling Interval** — The interval at which PROMON samples data, in seconds.
- **APW queue delay** — The value of the Page Writer Queue Delay (`-pwqdelay`) startup parameter. `-pwqdelay` is a self-tuning parameter. When the database encounters heavy loads, `-pwqdelay` decreases its value so that the APW writes more often. When the demand on the database lessens, `-pwqdelay` increases its value so that the APW writes less often. The default value, in milliseconds, for `-pwqdelay` is 100.
- **APW queue start** — The value of the Page Writer Queue Minimum (`-pwqmin`) startup parameter. Do not change this value unless directed to do so by Technical Support.
- **APW scan delay** — The value of the Page Writer Scan Delay (`-pwsdelay`) startup parameter. The default value, in seconds, of `-pwsdelay` is 1.
- **APW scan count** — The value of the Page Writer Scan (`-pwscan`) parameter. Do not change this value unless directed to do so by Technical Support.

- **APW write limit** — The value of the Page Writer Maximum Buffers (`-pwwmax`) parameter. Do not change this value unless directed to do so by Technical Support.
- **BIW scan delay** — The value of the BI Writer Scan Delay startup parameter (`-bwdelay`). The default value, in seconds, is 0.
- **Group commit delay** — The value of the Group Delay startup parameter (`-groupdelay`). The default value, in seconds, is 10.
- **Sort user lists** — Toggles the user list sort between sorting by user number or tenant Id
- **Tenant filter for user lists** — Allows you to filter by one or many tenants using the **Currently Connect Tenants screen**.

25

PROUTIL Utility

This chapter describes the OpenEdge database administration utility PROUTIL.

For details, see the following topics:

- PROUTIL utility syntax
- PROUTIL 2PHASE BEGIN qualifier
- PROUTIL 2PHASE COMMIT qualifier
- PROUTIL 2PHASE END qualifier
- PROUTIL 2PHASE MODIFY qualifier
- PROUTIL 2PHASE RECOVER qualifier
- PROUTIL AUDITARCHIVE qualifier
- PROUTIL AUDITLOAD qualifier
- PROUTIL AUDITRECONFIG qualifier
- PROUTIL BIGROW qualifier
- PROUTIL BULKLOAD qualifier
- PROUTIL BUSY qualifier
- PROUTIL CHANALYS qualifier
- PROUTIL CODEPAGE-COMPILER qualifier
- PROUTIL CONV1112 qualifier

- PROUTIL CONVCHAR qualifier
- PROUTIL CONVFILE qualifier
- PROUTIL DATAMOVE qualifier
- PROUTIL DBANALYS qualifier
- PROUTIL DBAUTHKEY qualifier
- PROUTIL DBIPCS qualifier
- PROUTIL DBRESTRICT qualifier
- PROUTIL DEALLOCATE qualifier
- PROUTIL DESCRIBE qualifier
- PROUTIL DISABLEAUDITING qualifier
- PROUTIL DISABLEAUTHGATEWAY qualifier
- PROUTIL DISABLEB2 qualifier
- PROUTIL DISABLECDC qualifier
- PROUTIL DISABLEENCRYPTION qualifier
- PROUTIL DISABLEJTA qualifier
- PROUTIL DISABLEKEYEVENTS qualifier
- PROUTIL DISABLEMULTITENANCY qualifier
- PROUTIL DISABLEREADONLYPARTITIONS qualifier
- PROUTIL DISABLETABLEPARTITIONING qualifier
- PROUTIL DISPTOSSCREATELIMITS qualifier
- PROUTIL DUMP qualifier
- PROUTIL DUMPSPECIFIED qualifier
- PROUTIL ENABLEAUDITING qualifier
- PROUTIL ENABLEAUTHGATEWAY qualifier
- PROUTIL ENABLEB2 qualifier
- PROUTIL ENABLECDC qualifier
- PROUTIL ENABLEENCRYPTION qualifier
- PROUTIL ENABLEJTA qualifier
- PROUTIL ENABLEKEYEVENTS qualifier
- PROUTIL ENABLELARGEKEYS qualifier
- PROUTIL ENABLEMULTITENANCY qualifier
- PROUTIL ENABLEREADONLYPARTITIONS qualifier
- PROUTIL ENABLESTOREDPROC qualifier

- PROUTIL ENABLETABLEPARTITIONING qualifier
- PROUTIL EPOLICY INFO qualifier
- PROUTIL EPOLICY MANAGE qualifier
- PROUTIL EPOLICY SCAN qualifier
- PROUTIL EPOLICY VIEW qualifier
- PROUTIL HOLDER qualifier
- PROUTIL IDXACTIVATE qualifier
- PROUTIL IDXANALYS qualifier
- PROUTIL IDXBUILD qualifier
- PROUTIL IDXCHECK qualifier
- PROUTIL IDXCOMPACT qualifier
- PROUTIL IDXDEACTIVATE
- PROUTIL IDXFIX qualifier
- PROUTIL IDXMOVE qualifier
- PROUTIL INCREASETO qualifier
- PROUTIL IOSTATS qualifier
- PROUTIL LOAD qualifier
- PROUTIL MTIDXBUILD qualifier
- PROUTIL MVSCH qualifier
- PROUTIL PARTITIONMANAGE ALTER qualifier
- PROUTIL PARTITIONMANAGE COPY qualifier
- PROUTIL PARTITIONMANAGE MERGE qualifier
- PROUTIL PARTITIONMANAGE RENAME qualifier
- PROUTIL PARTITIONMANAGE SPLIT qualifier
- PROUTIL PARTITIONMANAGE TRUNCATE qualifier
- PROUTIL PARTITIONMANAGE VIEW qualifier
- PROUTIL RCODEKEY qualifier
- PROUTIL SETAREACREATELIMIT qualifier
- PROUTIL SETAREATOSSLIMIT qualifier
- PROUTIL SETLOBCREATELIMIT qualifier
- PROUTIL SETLOBTOSSLIMIT qualifier
- PROUTIL SETTABLECREATELIMIT qualifier
- PROUTIL SETTABLETOSSLIMIT qualifier

- PROUTIL TABANALYS qualifier
- PROUTIL TABLEMOVE qualifier
- PROUTIL TPIDXBUILD qualifier
- PROUTIL TRUNCATE AREA qualifier
- PROUTIL TRUNCATE BI qualifier
- PROUTIL UPDATESCHEMA qualifier
- PROUTIL UPDATEVST qualifier
- PROUTIL VIEWB2 qualifier
- PROUTIL WBREAK-COMPILER qualifier
- PROUTIL WORD-RULES qualifier
- PROUTIL ZEROSTATS qualifier

PROUTIL utility syntax

PROUTIL performs various database operations. The qualifier you enter specifies the operation, and determines the required parameters you supply.

Syntax

```
proutil db-name -C qualifier
     [ [-userid username [-password passwrd] ] |
       [ -U username -P passwrd ]
     ]
     [-Passphrase]
```

Parameters

db-name

Specifies the database you are using.

-C

Specifies a particular utility or function when you use PROUTIL.

qualifier

Specifies the qualifier that you want to use. You can supply the qualifiers described in Table 97: PROUTIL utility qualifiers on page 633.

`-userid` *username* [`-password` *passwd*] | `-U` *username* `-P` *passwd*

A pair of username and password parameters are used together to support a common user authentication model to specify the user ID and password when connecting to an OpenEdge RDBMS. OpenEdge authenticates the user ID and password against the user accounts defined by the authentication system configured for the user's security domain.

Use username and password parameters to identify an authenticated database administrator.

If you do not specify a user ID and password for a database connection, OpenEdge assigns the OS process's user-id as the default user-id.

Note: You can also generate an encrypted password using the genpassword utility. For more information, see the encrypted password procedure in the Password (`-P`) connection parameter section of the *OpenEdge Deployment: Startup Command and Parameter Reference*. For more information on authentication systems and security domains, see *OpenEdge Getting Started: Identity Management*.

`-Passphrase`

For encryption-enabled databases only, specifies to prompt for a passphrase to authenticate the key store. Utilities that open the database must at a minimum authenticate as the key store user. Utilities that modify encryption policies (such as ENABLEENCRYPTION, DISABLEENCRYPTION, EPOLICY, TABLEMOVE, or INDEXMOVE) must authenticate as the key store admin. See Transparent Data Encryption on page 199 for more information on encryption and key store authentication.

Caution: If your database is enabled for transparent data encryption and configured for manual start, you must specify `-Passphrase` every time the database is opened.

Note: PROUTIL and its qualifiers support the use of internationalization startup parameters such as `-cpinternal codepage` and `-cpstream codepage`. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.

The table below lists all of the PROUTIL qualifiers. See the appropriate section for each of the qualifiers for more detailed information.

Table 97: PROUTIL utility qualifiers

Qualifier	Description
2PHASE BEGIN	Enables two-phase commit.
2PHASE COMMIT	Determines whether the coordinator database committed a transaction.
2PHASE END	Disables two-phase commit.
2PHASE MODIFY	Changes the nickname or priority of a database for two-phase commit.
2PHASE RECOVER	Commits or aborts limbo transactions for a database.
AUDITARCHIVE	Archives audit data from a database to a binary file.
AUDITLOAD	Loads audit data from a binary file into a database.

Qualifier	Description
AUDITRECONFIG	Reconstruct audit data and index areas.
BIGROW	Specifies the number of BI clusters available to a database.
BULKLOAD	Loads data files into a database.
BUSY	Determines whether the database is currently in use.
CHANALYS	Displays information about free chain blocks.
CODEPAGE- COMPILER	Compiles a conversion map file from text format to binary format.
CONV112	Converts an OpenEdge Release 11 database to an OpenEdge Release 12 database.
CONVCHAR	Converts a database's character sector identifies a database's character set (for undefined databases).
CONVFILE	Converts a text file from one character set to any other character set.
DATAMOVE	Moves table or sequence data from a multi-tenant source database to a multi-tenant target database for a tenant or group.
DBANALYS	Displays statistical information about index, record, and free chain blocks.
DBAUTHKEY	Sets an authorization key for a database.
DBIPCS	Displays status information for shared-memory segments attached by all OpenEdge databases on the system.
DBRESTRICT	Enables or disables write access for processes other than a specified utility.
DEALLOCATE	Deallocate a multi-tenant table from a database enabled for multi-tenancy
DESCRIBE	Describes the enabled features of a database.
DISABLEAUDITING	Disables auditing for a database.
DISABLEB2	Removes an area from Alternate Buffer Pool processing.
DISABLEENCRYPTION	Disables transparent data encryption for a database.
DISABLEJTA	Disables database participation in distributed JTA transactions.
DISABLEKEYEVENTS	Disables the storage of key events for a database.
DISABLEMULTITENANCY	Disables the support of multi-tenant tables.

Qualifier	Description
DISABLEREADONLYPARTITIONS	Disables the ability to change a partition to a read-only state in a table-partitioned database.
DISABLETABLEPARTITIONING	Disables the support of table-partitioning tables.
DISPTOSSCREATELIMITS	Display the block toss and create limits for all tables and LOBS in a specified area.
DUMP	Performs a binary dump, which is generally faster that an ASCII dump.
DUMPSPECIFIED	Performs a selective binary dump and allows you to dump by field value.
ENABLEAUDITING	Enables auditing for a database.
ENABLEB2	Designates an area for Alternate Buffer Pool processing.
ENABLEENCRYPTION	Enables transparent data encryption for a database.
ENABLEJTA	Enables the database to participate in distributed JTA transactions.
ENABLEKEYEVENTS	Enables the storage of key events for a database.
ENABLELARGEKEYS	Enables large key support for a database migrated to this release.
ENABLEMULTITENANCY	Enables support of multi-tenant tables.
ENABLEREADONLYPARTITIONS	Enables a table-partitioned database with the ability to change a partition to a read-only state.
ENABLESTOREDPROC	Enables stored procedures and triggers for 64-bit databases created prior to this release.
ENABLETABLEPARTITIONING	Enables support of table-partitioning tables.
EPOLICY INFO	Displays the valid ciphers for the database master key and object-level encryption and their command line cipher number.
EPOLICY MANAGE	Manages encryption policies for a specified database object, or the autostart or key store status.
EPOLICY SCAN	Reports on the encryption policies for the specified database object, or the autostart or key store status.
EPOLICY VIEW	Views the encryption policies for the specified database object, or the autostart or key store status.
HOLDER	Determines whether the database is currently in use in single-user mode, multi-user mode, or by a utility.
IDXACTIVATE	Activates an inactive index.

Qualifier	Description
IDXANALYS	Displays information on index blocks.
IDXBUILD	Consolidates index records to use disk space as efficiently as possible; activates deactivated indexes in the database. Repairs corrupted indexes in the database.
IDXCHECK	Checks database indexes to determine whether an index is corrupt, and if it is, diagnoses the problem.
IDXCOMPACT	Performs index compaction online and increases space utilization of the index block to a specified compacting percentage.
IDXDEACTIVATE	Deactivates an active index or indexes for databases enabled for multi-tenancy or table partitioning.
IDXFIX	Checks database records and indexes to determine whether an index is corrupt or a record has a missing or incorrect index; you can scan the database, the indexes, or both. IDXFIX will also repair corrupted indexes.
IDXMOVE	Moves an index from one application data area to another while the database remains online.
INCREASETO	Increase certain server startup parameters when the database is online.
IOSTATS	Provides current statistics for active databases. The statistics include buffered, unbuffered, and logical I/O database operations. The statistics are cumulative from database startup.
LOAD	Performs a binary load.
MTIDXBUILD	Rebuild an index for a tenant or group of a multi-tenant database.
MVSCH	Moves schema after a database conversion.
PARTITIONMANAGE ALTER	Changes the state of partitions in a table-partitioned database.
PARTITIONMANAGE COPY	Copies data into a partitioned table of a table-partitioned database from a temporary source.
PARTITIONMANAGE MERGE	Merges partitions of a table-partitioned database.
PARTITIONMANAGE RENAME	Renames a partition of a table-partitioned database.
PARTITIONMANAGE SPLIT	Splits partitions of a table-partitioned database.
PARTITIONMANAGE TRUNCATE	Truncates partitions of a table-partitioned database.
PARTITIONMANAGE VIEW	Displays information about partitions of a table-partitioned database.

Qualifier	Description
RCODEKEY	Inserts the authorization key into existing CRC-based r-code.
SETAREACREATELIMIT	Sets the create limit for the specified area.
SETAREATOSSLIMIT	Sets the toss limit for the specified area.
SETBLOBCREATELIMIT	Sets the create limit for the specified BLOB object. This command is deprecated and replaced by SETLOBCREATELIMIT
SETBLOBTOSSLIMIT	Sets the toss limit for the specified BLOB object. This command is deprecated and replaced by SETLOBTOSSLIMIT
SETLOBCREATELIMIT	Sets the create limit for the specified LOB object.
SETLOBTOSSLIMIT	Sets the toss limit for the specified LOB object.
SETTABLECREATELIMIT	Sets the create limit for the specified table.
SETTABLETOSSLIMIT	Sets the toss limit for the specified table.
TABANALYS	Displays information about the degree of fragmentation for each table in a database.
TABLEMOVE	Moves a table and optionally its associated indexes from one storage area to another while the database remains online.
TPIDXBUILD	Rebuild indexes of a partition in a table-partitioned database.
TRUNCATE AREA	Truncates application data storage areas in the specified database; use this qualifier before you remove storage areas and extents.
TRUNCATE BI	Performs three functions: Uses the information in the before-image (BI) files to bring the database up to date, waits to verify that the information has been successfully written to the disk, then truncates the before-image file to its original length. Sets the BI cluster size using the Before-image Filename (`-bi`) parameter. Sets the BI block size using the Before-image Block Size (`-biblocksize`) parameter.
UPDATESCHEMA	Loads the specified database with the most recent meta-schema.
UPDATEVST	Loads the specified database with the most recent virtual system tables.
VIEWB2	Lists the assignment of areas and objects with respect to Alternate Buffer Pool processing
WBREAK- COMPILER	Compiles the word-break table.
WORD-RULES	Compiles a word-rules file.

PROUTIL 2PHASE BEGIN qualifier

Enables two-phase commit for an OpenEdge database.

Syntax

```
proutil db-name -C 2phase begin  [ -crd | -tp nickname] . . .
```

Parameters

db-name

Specifies the database you are using.

-crd

Specifies that the database receive top priority when PROUTIL assigns a coordinator database. If you do not specify a database, PROUTIL randomly assigns a coordinator database from the available databases.

Specifying a single coordinator database can greatly simplify the process of resolving limbo transactions because PROUTIL does not have to consult several different coordinating databases.

-tp *nickname*

Identifies a unique nickname or alternate name that PROUTIL uses to identify the coordinator database. If you do not specify a nickname, PROUTIL automatically assigns the name of the database without the .db extension as the nickname.

Specifying a nickname simplifies resolving limbo transactions when two databases have the same pathname but are on different machines.

Notes

- You cannot use the PROUTIL 2PHASE BEGIN qualifier against a database that is online.
- A TL (Transaction Log) area is required to hold the transaction log data generated when two-phase commit is in use. You must create and maintain a TL area for your database in order to use two-phase commit.

PROUTIL 2PHASE COMMIT qualifier

Determines if the coordinator database committed a transaction.

Syntax

```
proutil db-name -C 2phase commit tr-number
```

Parameters

db-name

Specifies the coordinator database.

tr-number

Specifies the number of the transaction where you want to check the commit status.

If the coordinator committed the transaction, PROUTIL displays a message similar to the following:

```
Transaction 768 has committed. (2048)
```

Once you determine that the coordinator database committed the transaction, you must also commit the transaction on the database where the limbo transaction occurred. If the coordinator did not commit the transaction, you must terminate the transaction on the database where the limbo transaction occurred.

PROUTIL 2PHASE END qualifier

Disables two-phase commit for a database.

Syntax

```
proutil db-name -C 2phase end
```

Parameters

db-name

Specifies the name of the database where you want to disable two-phase commit.

PROUTIL 2PHASE MODIFY qualifier

Changes the nickname or priority of a database for two-phase commit.

Syntax

```
proutil db-name -C 2phase modify [ -crd | -tp nickname] . . .
```

Parameters

db-name

Specifies the database you are using.

-crd

Determines whether or not the database can serve as a coordinator database. If you specify -crd against a database that is a candidate for coordinator database, it is no longer a candidate. If you specify -crd against a database that is not a candidate, it becomes one.

-tp *nickname*

Identifies a new nickname for the coordinator database. Specifying a nickname simplifies resolving limbo transactions when two databases have the same pathname but are on different machines.

PROUTIL 2PHASE RECOVER qualifier

Commits or aborts limbo transactions for a database.

Syntax

```
proutil db-name -C 2phase recover
```

Parameters

db-name

Specifies the database with the limbo transaction.

When you run this command against a database with limbo transactions, PROUTIL displays a message similar to the following:

```
Commit transaction 760, on coordinator sports1 #768 (y to commit/n to
abort)? (2039)
```

If you type y, PROUTIL commits the transaction. If you type n, PROUTIL aborts the transaction.

PROUTIL AUDITARCHIVE qualifier

Archives records from your audit data tables to a binary file.

Syntax

```
proutil db-name -C auditarchive [date-range]
    [-recs num-recs] [-nodelete] [-checkseal]
    [-directory directory-name]
    [-userid user-id[-password passwd] ]
    [ -Cipher cipher-id]
```

Parameters

db-name

Specifies the database whose audit data you are archiving.

date-range

When specified, *date-range* limits the archived records as shown in the following table:

If your date range is . . .	Then the data archived is . . .
Blank	All audit data records in the database at the time the archive process begins.
One date	All audit data records in the database with a date equal to or earlier than the specified date.
Two dates, separated by a space	All audit data records in the database with times that fall between the two dates specified, inclusively. You must first specify the earlier date, followed by the later date.

Specify *date-range* with the format "*mm-dd-yyyy hh:mm:ss.sss+hh:mm*". You must include the quotation marks. For example, to specify 5:00 P.M. on September 7, 2005 EDT, enter "09-07-2005 17:00:00.000-04:00".

-recs *num-recs*

Specifies the number of records to delete in a single transaction. AUDITARCHIVE executes a transaction for every *num-recs* records it deletes. The default for *num-recs* is 100, and the maximum is your current Lock Table Entries (-L) setting. The multi-user default for -L is 8192.

-nodelete

Specifies that AUDITARCHIVE copy the audit data records to the archive file, leaving the audit data in the database. If not specified, AUDITARCHIVE will delete the audit data records as they are archived.

-checkseal

Specifies that AUDITARCHIVE verify that the seal of each audit data record matches the database MAC key prior to writing it to the audit archive file. This option requires that the audit policies in effect when writing the audit data records have specified a Data Security Level of DB Passkey. For more information on creating audit policies and data security, see *OpenEdge Getting Started: Core Business Services - Security and Auditing.*

-directory *directory-name*

Specifies the directory where the audit archive file is written. If you do not specify a value for *directory-name*, the file is written to your current working directory. If an audit data archive file for your database already exists in the output directory, you are prompted to confirm if you wish to proceed and overwrite the existing file, or exit the utility.

If you are purging your audit data and do not want the data written to the archive file, you must explicitly specify the "bit bucket" (/dev/null) as the output destination in *directory-name*.

-userid *username*

Specifies the user name of the privileged user. If you are using the _User table, *username* must be defined there, and you must include the -userid parameter to identify the user. If you are not using the _User table, use of -userid is optional; in this scenario, *username* specifies a local operating system login ID. Regardless of the source, *username* must have the Audit Data Archiver privilege in order to run AUDITARCHIVE.

-password *passwd*

Specifies the password for *username*. If you are using the _User table, *passwd* is the password defined for the user there. The value of *passwd* can be clear text or encrypted. If you are not using the _User table, and have specified a local operating system ID with -userid, *passwd* must be the encrypted value of the DB Pass key. See *OpenEdge Getting Started: Core Business Services - Security and Auditing* or the Data Administration online Help for information on the DB Pass Key. For information on encrypting a password, see the description of the genpassword utility in *OpenEdge Getting Started: Installation and Configuration.*

If you do not supply a password corresponding to *username*, the utility will prompt for it.

-Cipher *cipher-id*

For an Enterprise database enabled for Transparent Data Encryption, specify -Cipher 11 or -Cipher 12 to encrypt the output of the PROUTIL AUDITARCHIVE with a password-based encryption cipher (PBE). You are prompted to enter the passphrase for the cipher. You must remember this passphrase; you will be prompted to enter the passphrase when loading the audit archive.

OpenEdge versions prior to Version 12.0 permitted the use of -Cipher 6 for an Enterprise database enabled for Transparent Data Encryption. In Version 12.0, if AUDITARCHIVE specifies -Cipher 6, the command will continue with a warning:

```
 Warning: The minimum allowed cipher for this operation is AES128_CBC_PBE.(19384)

Substituting cipher AES128_CBC_PBE for requested DES_CBC_PBE. (19385)
```

AUDITARCHIVE archives audit data from the _aud_audit-data and _aud-audit-data-value tables. It also copies data from the _db-detail, _client-session, and _aud-event tables. The data archived is written to one file, named *db-name*.abd so that the data from all the tables is treated as a unit. The file is sealed with a time stamp and the MAC seal currently stored in the database.

Archiving audit data is an auditable event. If your active audit policy is configured to audit the execution of AUDITARCHIVE, then every time it runs an audit event will be generated. The audit data record that audits the archive event will not be deleted from the database when AUDITARCHIVE runs.

Notes

- The Audit Archive privilege is required to run AUDITARCHIVE.
- AUDITARCHIVE runs against a database that is online or offline.
- Multiple instances of AUDITARCHIVE can simultaneously run against the same database, however their output destinations must be unique directory specifications. If the data from multiple archives contains duplicate records, AUDITLOAD ignores the duplicates and continues to load the data without error.
- If an archive file exists in the specified output directory, AUDITARCHIVE will not overwrite it. AUDITARCHIVE issues a warning message and provides you with the choice of overwriting the existing file or exiting.

- If your archive file reaches the maximum file size for your operating system, a second file, *db-name*2.abd, will be written.
- When archiving audit data encrypted with Transparent Data Encryption, the output of the audit archive is **not** encrypted by default. Encrypt the archive file (container security) by specifying -Cipher 11 or -Cipher 12and entering a passphrase for the PBE cipher. You will be prompted to enter the passphrase when loading the contents of the archive file.

PROUTIL AUDITLOAD qualifier

Loads audit data into your database from your audit archive file.

Syntax

```
proutil db-name -C auditload archive-file-name
    [-recs num-recs] [-checkseal]
    [-userid user-id [-password passwd] ]
    [-Cipher cipher-id]
```

Parameters

db-name

Specifies the database where the archived data is being loaded.

archive-file-name

Specifies the audit archive file to load.

-recs *num-recs*

Specifies the number of records to load in a single transaction. AUDITLOAD executes a transaction for every *num-recs* records it loads. The default for *num-recs* is 100, and the maximum is your current Lock Table Entries (-L) setting. The multi-user default for -L is 8192.

-checkseal

Specifies that AUDITLOAD verify that the seal of each audit data record matches the database MAC key prior to loading it into the archive database. This option requires that the audit policies in effect when the audit data records were generated, specified a Data Security Level of DB Passkey. For more information on creating audit policies and data security, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*. During the AUDITLOAD, the _db-detail record containing the database MAC key is loaded into the archive database before any of the audit data records, making it available for the verification process.

-userid *username*

Specifies the user name of the privileged user. If you are using the _User table, *username* must be defined there, and you must include the -userid parameter to identify the user. If you are not using the _User table, use of -userid is optional; in this scenario, *username* specifies a local operating system login ID. Regardless of the source, *username* must have the Audit Data Archiver privilege in order to run AUDITLOAD.

`-password` *`passwd`*

Specifies the password for *`username`*. If you are using the `_User` table, *`passwd`* is the password defined for the user there. The value of *`passwd`* can be clear text or encrypted. If you are not using the `_User` table and have specified a local operating system ID with `-userid`, *`passwd`* must be the encrypted value of the DB Pass key. See *OpenEdge Getting Started: Core Business Services - Security and Auditing* or the Data Administration online Help for information on the DB Pass Key. For information on encrypting a password, see the description of the `genpassword` utility in *OpenEdge Getting Started: Installation and Configuration.*

If you do not supply a password corresponding to *`username`*, the utility will prompt for it.

`-Cipher` *`cipher-id`*

For an Enterprise database enabled for Transparent Data Encryption, specify `-Cipher 11` or `-Cipher 12` to indicate that the contents of the archive file being loaded are encrypted. The specified *`cipher id`* should match the *`cipher id`* specified by the PROUTIL AUDITARCHIVE command at the time the archive file was created. You are prompted to enter the passphrase that was specified when the archive file was created.

Note: OpenEdge versions prior to Version 12.0 permitted the use of `-Cipher 6` for an Enterprise database enabled for Transparent Data Encryption. PROUTIL AUDITLOAD in OpenEdge Version 12.0 still permits `-Cipher 6` to be specified to load archive files created by previous versions of OpenEdge.

Prior to beginning the load, AUDITLOAD verifies the data seal of the archive file. If AUDITLOAD encounters a duplicate record, it is ignored without error.

Loading archived audit data is an auditable event. If the active audit policy on the archive database is configured to audit AUDITLOAD, then every time it runs, an audit event will be generated.

Notes

- The Audit Data Archiver privilege is required to run AUDITLOAD.
- AUDITLOAD runs against a database that is online as well as offline.
- Multiple instances of AUDITLOAD can simultaneously run against the same database.
- AUDITLOAD generates and updates index information for each record at the time it is loaded.
- If the contents of the archive file was encrypted with `-Cipher 11` or `-Cipher 12` when created, you must specify `-Cipher 11` or `-Cipher 12` for PROUTIL AUDITLOAD to indicate that the file is encrypted. You are prompted to enter the passphrase entered when the file was created. You cannot load the contents without the passphrase.
- When loading the contents of an encrypted archive file, the loaded records are not encrypted unless there is an active encryption policy on the table before the load begins. See PROUTIL EPOLICY MANAGE qualifier on page 705 or Transparent Data Encryption on page 199 for more information on creating an encryption policy.

PROUTIL AUDITRECONFIG qualifier

Reconstructs the audit data and index areas of an audit-enabled database.

Syntax

```
proutil db-name -C auditreconfig [ tablearea table-area-name]
    [ indexarea index-area-name]
    [-userid username [-password passwd] ]
```

Parameters

db-name

Specifies the database where you want to reconfigure your audit data and index areas.

tablearea table-area-name

Name of the Type II area where auditing data tables are to be located.

indexarea index-area-name

Name of the Type II area where auditing indexes are to be located.

userid *username*

Specifies the user name of the privileged user. If you are using the _User table, *username* must be defined there, and you must include the -userid parameter to identify the user. If you are not using the _User table, use of -userid is optional; in this scenario, *username* specifies a local operating system login ID. Regardless of the source, *username* must have the Audit Data Archiver privilege in order to run AUDITRECONFIG.

-password *passwd*

Specifies the password for *username*. If you are using the _User table, *passwd* is the password defined for the user there. The value of *passwd* can be clear text or encrypted. If you are not using the _User table and have specified a local operating system ID with -userid, *passwd* must be the encrypted value of the DB Pass key. See *OpenEdge Getting Started: Core Business Services - Security and Auditing* or the Data Administration online Help for information on the DB Pass Key. For information on encrypting a password, see the description of the genpassword utility in *OpenEdge Getting Started: Installation and Configuration.*

If you do not supply a password corresponding to *username*, the utility will prompt for it.

AUDITRECONFIG provides Audit Data Archivers the ability to reconfigure audit data and index areas while auditing is enabled. You can move the audit data tables and indexes to new areas, or you can purge and reuse your current areas.

When reconfiguring your audit data and index areas, you can:

- Split or combine index and data areas
- Set new records per block and blocks per cluster values in new areas
- Purge all data from existing audit areas by specifying the existing areas in the AUDITRECONFIG command
- Reclaim disk space by deleting your old audit data and or index areas after configuring new areas with PROSTRCT REMOVE.

If data exists in the audit data area it is destroyed by AUDITRECONFIG. You are prompted to acknowledge this fact before the reconfiguration executes.

See Reconfiguring audit areas on page 189 for examples.

Notes

- You must have Audit Archiver privilege to run AUDITRECONFIG.
- Your database must be offline to execute PROUTIL AUDITRECONFIG.
- If auditing is disabled, you can execute AUDITRECONFIG if the audit data tables are empty.
- AUDITRECONFIG destroys all data in your current audit data and index areas. Preserve your auditing data by archiving with PROUTIL AUDITARCHIVE prior to executing PROUTIL AUDITRECONFIG. See PROUTIL AUDITARCHIVE qualifier on page 640 for more information on archiving audit data.

PROUTIL BIGROW qualifier

Specifies the number of BI clusters to preformat for a database.

Syntax

```
proutil db-name -C bigrow n [ -r ]
```

Parameters

db-name

Specifies the database where you want to create BI clusters.

n

Specifies the number of BI clusters to create for the specified database.

-r

Nonreliable I/O startup parameter; turns off reliability.

Notes

- You can calculate the number of BI clusters for a database by dividing the BI file physical size by the BI cluster size. For example, a database BI file with a BI cluster size of 128K and a physical size of 917,504 has 7 BI clusters.
- By default, four BI clusters are created at startup. Any BI clusters specified by *n* are added to the original four.
- The database must be offline to use the BIGROW qualifier.
- Use the BIGROW qualifier immediately after truncating the BI file.
- See Managing Performance on page 275 for information about using the BIGROW qualifier.

PROUTIL BULKLOAD qualifier

Loads text data files into a database.

Note: The bulk loader works only with OpenEdge databases. Some non-OpenEdge databases offer a similar bulk-loading tool. If such a tool is not available, use the standard load option in the Data Administration tool or Data Dictionary. BULKLOAD deactivates indexes on the tables being loaded. Indexes should be rebuilt before accessing data.

Syntax

```
proutil db-name [ -yy n] -C BULKLOAD fd-file
   [ tenant tenant-name | group group-name
   | partition partition-name | composite initial ]
   [ datadir directory-name] [-B n ]
```

Parameters

db-name

Specifies the database you are using.

-yy *n*

Century Year Offset startup parameter; *n* specifies a four-digit year (1900, for example) that determines the start of a 100-year period in which any two-digit DATE value is defined. The default is 1950, but the -yy n value must match the -yy n value used when the data was dumped.

fd-file

Identifies the bulk loader description file you are using.

tenant *tenant-name*

Specifies that the BULKLOAD operation is to load data into the partition owned by *tenant-name*. If *tenant-name* is not valid, BULKLOAD exits with an error. Specifying tenant is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the BULKLOAD operation is to load data into the partition owned by *group-name*. If *group-name* is not valid, BULKLOAD exits with an error. Specifying group is only allowed on databases enabled for multi-tenancy.

partition *partition-name*

Specifies that the BULKLOAD operation is to load data for the table partition *partition-name*. If *partition-name* is not valid, BULKLOAD exits with an error. Specifying partition is only allowed on partitioned tables in databases enabled for table partitioning.

`composite initial`

Specifies that the BULKLOAD operation is to load data for the composite initial partition of the specified table. Specifying `composite initial` is only allowed on partitioned tables in databases enabled for table partitioning.

`datadir` *`directory-name`*

Specifies the location of the data (`.d`) files. If not specified, BULKLOAD looks in the current directory.

`-B` *`n`*

Blocks in Database Buffers startup parameter; `n` specifies the number of blocks.

Notes

- You must create the bulk loader description file and load the data definitions (`.df`) files for the database before you can run the Bulk Loader utility. See Dumping and Loading on page 411 for information about creating the bulk loader description file and loading data definitions files.
- You cannot use BULKLOAD to load protected audit data. For more information on auditing and utility security and restrictions, see Auditing impact on database utilities on page 193
- If you do not specify `group` or `tenant` for a multi-tenant table, BULKLOAD loads the data into the default partition.
- For a partitioned table, if you specify a `partition` or `composite initial`, BULKLOAD loads data for that partition only, and reports on the number of records skipped that do not belong in the specified partition. If you do not specify a partition, but are loading a partitioned table, BULKLOAD loads each record into the appropriate partition of the table.
- When loading a partitioned table, if any partitions cannot be written to, a warning is issued, and the load proceeds; if no partitions can be written to, BULKLOAD exits with an error message.

PROUTIL BUSY qualifier

Determines if the database is currently in use.

Syntax

```
proutil db-name -C busy
```

Parameters

`db-name`

Specifies the database you are using.

Determining if the database is in use is useful before performing a database backup or shutting down a database. A database is in use if it is being used in single-user mode, multi-user mode, or by an OpenEdge utility program.

To get the database status, you must run PROUTIL with the BUSY qualifier and test the command return code in a UNIX script or Windows batch file. The return codes for the BUSY qualifier are shown in the following table:

Return code	Description
0	Database is not in use
6	Database is in use
64	Database is in process of starting up

Example

This example shows how you might use the BUSY qualifier on UNIX in a script that tests if the database is busy:

```
proutil mydb -C busy

if [ $? != 0 ]
then
  echo
  echo "Do you want to use 'proshut' to force users off\
    the system?"
  read ans
  if [ "$ans" = y ]
  then
    proshut -by mydb
  else
    echo "Backup will not be performed."
    exit
  fi
fi
echo "Beginning backup."

# Backup procedure
```

PROUTIL CHANALYS qualifier

Displays information about the index delete change, and the free and record manager (RM) chains. It calculates the number of blocks found both in the free chain and in the RM chain.

Syntax

```
proutil db-name -C chanalys [ area area-name ]
     [ tenant tenant-name | group group-name | shared ]
     [ -scan ] [-csoutput [ -fieldsep sep-value]
     [ -csfilePrefix pref-name ] [ -verbose ] ]
```

Parameters

db-name

Specifies the database you are using.

`area` *`area-name`*

Specifies that CHANALYS is to be performed on the *`area-name`* area only.

`tenant` *`tenant-name`*

Specifies that CHANALYS is to be filtered to only provide information related to objects that are owned by *`tenant-name`*. If *`tenant-name`* is not valid, CHANALYS exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *`group-name`*

Specifies that CHANALYS is to be filtered to only provide information related to objects that are owned by *`group-name`*. If *`group-name`* is not valid, CHANALYS exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`shared`

Specifies that CHANALYS is to be filtered to only provide information related to shared objects. Specifying `shared` is only allowed on databases enabled for multi-tenancy.

`-scan`

Specifies that CHANALYS is to perform RM-chain analysis with minimal locking (does not turn on table locks). This option should be used when you are concerned that the locking required for a full analysis will be a significant performance impact. The information displayed may be incomplete because the chain can be changing while the analysis is being performed.

The following aspects of chain analysis are omitted with this option:

- Verification that each block is below the high-water mark.
- Verification that each block appears on the chain only once.
- Verification that the object block's reported count matches the chain length (values may not agree because of the reduced locking).
- Verification that the block is the correct type (RM block for RM chain, index block for index chain).
- Verification of cluster pointers.

`-csoutput`

Specifies that CHANALYS output the analysis data to text files in addition to the screen. The result is that the following files are created:

- *`db-name`*`.ch.txt` — For chain analysis output
- *`db-name`*`.block.txt` — For block analysis output

For more information about the contents of the files, see the Database analysis output on page 667.

`-fieldsep` *`sep-value`*

When `-csoutput` is specified, *`sep-value`* indicates the value to separate the columns of output. A space is used by default. Accepted values for *`sep-value`* are:

- Any ASCII character between 0x21 and 0x7E
- `sp` — to denote a space (the default)
- `tab` — to denote a the TAB character (`<\t>` or ASCII character 0x09)

`-csfilePrefix` *`pref-name`*

When `-csoutput` is specified, *`pref-name`* indicates a file name prefix. The database name is used by default. All remaining elements of the file name are not customizable.

`-verbose`

When `-csoutput` is specified, the first row of the output files contains header names for each column of output.

Output format

The chain analysis report appears in internal storage order (as stored in the object block for the area) with a column to identify **Partition/Tenant/Group**. The format of this column is as follows: *type*:*name*:*id* where *type* is **P** for Partition, **T** for Tenant, or **G** for Group; *name* is the partition name, tenant name, or group name; and *id* is the partition-id, tenant-id, or group-id. The column is blank if there is no partition, tenant, or group.

Example

The following is a sample of PROUTIL CHANALYS output.

```
CHAIN ANALYSIS FOR AREA "tpArea2": 11
------------------------------------------------------------

FREE CLUSTER CHAIN ANALYSIS
---------------------------

0 cluster(s) found in the free cluster chain.


FREE CHAIN ANALYSIS
-------------------

          LIST OF FREE CHAIN BLOCKS
                 next
        dbkey    free
        448      0

Number of       Object     Object                                        Partition/Tenant/Group
Blocks           Type
-------------------------------------------------------------------------------------------------
1                Master     --:0
0                Table      PUB.tpOrder-Line:1
          LIST OF FREE CHAIN BLOCKS
                 next
        dbkey    free
        1472     0
1                Index      PUB.tpOrder-Line.tpOrder-Line:8
          LIST OF FREE CHAIN BLOCKS
                 next
        dbkey    free
        1856     1920
        1920     1984
        1984     0
3                Index      PUB.tpInvoice.tpCustNum:15
0                Table      PUB.tpOrder:5
          LIST OF FREE CHAIN BLOCKS
                 next
        dbkey    free
        2880     2944
        2944     3008
        3008     0
3                Index      PUB.tpItem.tpItem-Num:23
```

```
          LIST OF FREE CHAIN BLOCKS
                next
        dbkey   free
        3392    3456
        3456    3520
        3520    0
3                Index     PUB.tpItem.tpCat-Description:24
          LIST OF FREE CHAIN BLOCKS
                next
        dbkey   free
        3904    3968
        3968    4032
        4032    0
3                Index     PUB.tpItem.tpItem-Name:25
          LIST OF FREE CHAIN BLOCKS
                next
        dbkey   free
        4416    4480
        4480    4544
        4544    0
3                Index     PUB._Partition-Policy-Detail._Partition-Internal-Value:-1558
                                                                        P:PUB.tpInvoice:3

          LIST OF FREE CHAIN BLOCKS
                next
        dbkey   free
        4928    4992
        4992    5056
        5056    0
3                Index     PUB._Partition-Policy-Detail._Partition-Internal-Value:-1558
                                                                    P:PUB.tpLargeField:4
0                Table     PUB.tpCustomer:2                        P:CustomerQtr1:1
0                Blob      PUB.tpCustomer.CustBlob1:5              P:CustomerQtr1:1
0                Blob      PUB.tpCustomer.CustBlob2:6              P:CustomerQtr1:1
0                Blob      PUB.tpCustomer.CustClob1:7              P:CustomerQtr1:1
          LIST OF FREE CHAIN BLOCKS
                next
        dbkey   free
        7488    7552
        7552    7616
        7616    0
3                Index     PUB.tpInvoice.tpInvoiceNum:14           P:InvoiceQtr1:1
0                Table     PUB.tpLargeField:4                      P:LrgFldQtr1:1

RM CHAIN ANALYSIS
-----------------------------

          LIST OF RM CHAIN BLOCKS
                free    # free
        dbkey   space   slots   hold
        89152   6519    22      0
        89216   8124    64      0
        89280   8124    64      0
        89344   8124    64      0
        89408   8124    64      0
        89472   8124    64      0
        89536   8108    64      0

Number of       Object     Object                                Partition/Tenant/Group
Blocks           Type
-------------------------------------------------------------------------------------------
7                Table     PUB.tpOrder-Line:1
          LIST OF RM CHAIN BLOCKS
                free    # free
        dbkey   space   slots   hold
        88128   5097    14      0
        88192   8124    64      0
        88256   8124    64      0
        88320   8124    64      0
```

```
        88384   8124    64      0
        88448   8124    64      0
        88512   8108    64      0
7                Table     PUB.tpOrder:5
          LIST OF RM CHAIN BLOCKS
                 free    # free
        dbkey   space   slots   hold
        92736   7027    62      0
        92800   8124    64      0
        92864   8124    64      0
        92928   8124    64      0
        92992   8124    64      0
        93056   8124    64      0
        93120   8108    64      0
        5568    228     44      1
        5440    242     40      1
        5504    226     40      1
        92672   699     51      0
11               Table     PUB.tpCustomer:2                                P:CustomerQtr1:1
          LIST OF RM CHAIN BLOCKS
                 free    # free
        dbkey   space   slots   hold
        65920   8124    64      0
        65856   8124    64      0
        65792   8112    63      1
        65664   8124    64      0
        65600   8124    64      0
        65536   8096    63      1
        42880   8124    64      0
         .
         .
         .
        76672   456     63      0
        37120   328     49      1
        31872   3697    20      1
        71296   316     51      1
133              Blob      PUB.tpCustomer.CustBlob1:5                      P:CustomerQtr1:1
          LIST OF RM CHAIN BLOCKS
                 free    # free
        dbkey   space   slots   hold
        84352   8112    63      1
        84224   8124    64      0
         .
         .
         .
        56768   319     55      1
        11200   332     56      1
        12992   319     55      1
311              Blob      PUB.tpCustomer.CustBlob2:6                      P:CustomerQtr1:1
          LIST OF RM CHAIN BLOCKS
                 free    # free
        dbkey   space   slots   hold
        31040   8112    63      1
        30784   8112    63      1
         .
         .
         .
        59264   337     61      1
        62016   305     61      1
        62272   337     61      1
159              Blob      PUB.tpCustomer.CustClob1:7                      P:CustomerQtr1:1
          LIST OF RM CHAIN BLOCKS
                 free    # free
        dbkey   space   slots   hold
        92544   8112    63      1
        91776   1418    63      0
         .
         .
         .
```

```
        93184   1418    63      0
        90240   1418    63      0
42               Table     PUB.tpLargeField:4                                  P:LrgFldQtr1:1

INDEX DELETE CHAIN ANALYSIS
---------------------------


Number of       Object     Object                                        Partition/Tenant/Group
Blocks           Type
-------------------------------------------------------------------------------------------
0                Index     PUB.tpOrder-Line.tpOrder-Line:8
0                Index     PUB.tpInvoice.tpCustNum:15
0                Index     PUB.tpItem.tpItem-Num:23
0                Index     PUB.tpItem.tpCat-Description:24
0                Index     PUB.tpItem.tpItem-Name:25
0                Index     PUB._Partition-Policy-Detail._Partition-Internal-Value:-1558
                                                                    P:PUB.tpInvoice:3
0                Index     PUB._Partition-Policy-Detail._Partition-Internal-Value:-1558
                                                                    P:PUB.tpLargeField:4
0                Index     PUB.tpInvoice.tpInvoiceNum:14           P:InvoiceQtr1:1
```

PROUTIL CODEPAGE-COMPILER qualifier

Compiles a conversion map file from text format to binary format.

Syntax

```
proutil -C codepage-compiler inputfile outputfile
```

Parameters

inputfile

Specifies the name of the text conversion map file. This file must follow a specific format.

outputfile

Specifies the name of the binary conversion map file.

A *conversion map file* is a code page description file that contains various tables of information and uses a specific format. For more information about conversion map files, see *OpenEdge Development: Internationalizing Applications*.

PROUTIL CONV1112 qualifier

Converts an OpenEdge Release 11 database to an OpenEdge Release 12 database.

Syntax

```
proutil db-name -C conv1112
```

Parameters

db-name

Specifies the OpenEdge Release 11 source database name.

CONV1112 must be performed when the database is offline, and cannot be interrupted. If there is an abnormal exit from CONV1112, you must revert to a backup of your OpenEdge Release 11 database.

Before performing the conversion, you must:

- Back up your OpenEdge Release 11 database using a Release 11 back up utility
- Verify that the database is not in use
- Truncate the BI file with Release 11
- Disable after-imaging with Release 11
- Disable two-phase commit with Release 11

Note: If you have OpenEdge Replication enabled, you must disable it before you can disable after-imaging. For information on Replication, see *OpenEdge Replication: User Guide*.

CONV1112 performs a pre-scan to check for conditions that prevent a complete conversion. If any conditions exist that would prevent a complete conversion are found, they are reported, and CONV1112 exits normally with no change to the database. You must correct all reported issues before CONV1112 can successfully complete. Conditions that cause the pre-scan to exit include:

- The BI file is not truncated
- The database is online
- The database is not an OpenEdge Release 11 database
- The database has Transparent Data Encryption (TDE) enabled, and an encryption policy using the current or previous RC4 cipher (Cipher ID 7) exists on any table, index, or LOB.

Note: In the event conversion is cancelled as the result of existing encryption using the RC4 cipher (Cipher ID 7), you will receive a message that the cipher is no longer supported. You must then either:

- Use the PROUTIL EPOLICY utility to remove the RC4 cipher (Cipher ID 7) encryption from the encrypted objects, and PROUTIL EPOLICY UPDATE to retire the cipher; or

- Use the PROUTIL EPOLICY utility to change the RC4 cipher (Cipher ID 7) to a stronger encryption cipher, and use the PROUTIL EPOLICY UPDATE and PROUTIL EPOLICY SCAN operations to re-encrypt the objects with the new policy and cipher.

These operations may be performed on either the existing version 11 database or on a backup created at the time of conversion.

During the conversion:

- If your database has after-imaging enabled, the conversion process disables it, and issues a message.
- If your database has auditing enabled, it is temporarily disabled during the conversion, but re-enabled before the conversion completes.

- If your database has encryption enabled, it is temporarily disabled, preventing access to the encryption cache and any encrypted data in the database. Encryption is re-enabled when the conversion completes, and access to the encryption cache and any encrypted data restored.
- If your database has Change Data Capture (CDC) enabled, it is temporarily disabled, preventing CDC cache allocation and loading. CDC is re-enabled when the conversion completes.
- All schema in your database is updated to the latest version, and any new schema tables are added.
- All existing 32-bit sequences are upgraded to 64-bit.

  Existing 32-bit sequences set to cycle at the limit, with the Unknown value (`?`) as the limit, are changed to cycle at the maximum or minimum value of a 32-bit integer (+/-2147483647), keeping the behavior of a cycling sequence consistent.

  Existing 32-bit sequences that do *not* have a cycle and the limit is the Unknown value (`?`), are changed to have an implied maximum or minimum of a 64-bit integer (+/-9223372036854775807).

  Caution: Verify any impact the sequence conversion may have on your application. Review your database sequence schema after the conversion completes, to verify initial and cycle settings, or take advantage of the 64-bit values.

- All VSTs in your database are recreated and updated to the latest version, and any new VSTs are added.
- New auditing events are populated

Notes

- CONV1112 must be run offline.
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and CONV1112 exits.
- As database conversion is a single-user operation, the caches for multi-tenancy and OE Authentication Gateway are not allocated or loaded. Conversion does not involve these features.
- After completing the conversion, back up your OpenEdge Release 12 database right away.

PROUTIL CONVCHAR qualifier

Converts a database's character set or identifies a database's character set for undefined databases.

```
proutil db-name -C convchar [ analyze | charscan | convert ]
                            [codepage] [character-list]
```

Parameters

db-name

Specifies the database you are converting.

`analyze`

Scans the database and displays the fields that would be converted using the `convert` function.

`charscan`

Searches every character field for the occurrence of any character from the provided character list and reports the table name, field name, record ID of a match, and the total number of occurrences. In addition, PROUTIL CONVCHAR CHARSCAN performs the same analysis that PROUTIL CONVCHAR ANALYSIS performs.

If invalid data is entered with the `charscan` option, PROUTIL CONVCHAR generates an error message and continues with the scan.

`convert`

Converts a database's character data to the target code page and labels the database.

Note: PROUTIL CONVCHAR CONVERT does not convert DBCODEPAGE CLOBs (character large objects) to the new code page; instead, it changes DBCODEPAGE CLOBs to COLUMN-CODEPAGE CLOBs with their existing settings.

`codepage`

Specifies the value of any single-byte, double-byte, or Unicode code page. Possible values are undefined or any (target) code page name for a conversion table in your conversion map file (by default, *`OpenEdge-install-dir`*`/convmap.cp`). If you specify a code page name, the PROUTIL CONVCHAR utility must find the appropriate conversion table in the conversion map file. If you specify undefined, no conversions take place and the database's code page name is changed to undefined.

If you specify a code page name against a database with a code page name of undefined, no conversions take place and the database's code page name is changed to the value you specify.

Note: When you change the code page of a database from "undefined" using `proutil dbname -C convchar convert codepage`, PROUTIL CONVCHAR generates warning messages. After you receive these messages, you might need to load the collation table for the code page. The collation table data definition files have a `.df` extension and are located in *`OpenEdge-install-dir`*`/prolang/`*`<language>`*.

When you specify a code page value with `analyze`, it scans the database and identifies the database's current code page encoding and the fields that require translation if you were to use the convert function with that code page specification.

When you use the code page parameter with `convert`, it specifies the code page to which you want to convert the database and specifies the code page name for the database.

`character-list`

Specifies a quoted string of comma-separated numbers in either hex or decimal. These numbers represent character values in the code page. Specify ranges with a minus sign (`"128 - 144"`).

Note: Quotes are not needed if there are no blanks within the list (`128-144,122`).

Hex values must begin with `0x`. For example, the syntax to run PROUTIL CONVCHAR with the decimal list `"128, 129, 130"` provided in hex is:

```
proutil sports2000 -C convchar charscan ibm850 "0x80, 0x81, 0x82"
```

Note also that hex and decimal values can be mixed. For example:

```
proutil sports2000 -C convchar charscan 1253 "128, 0xC2, 0x7f, 122"
```

If a range contains valid and invalid characters, PROUTIL CONVCHAR ignores the invalid character. For example:

```
proutil sports2000 -C convchar charscan 1253 "49854 - 50050"
```

Searches and returns the following:

```
Charscan searching for utf-8 character: 49854 0xc2be. (6570) Charscan
searching for utf-8 character: 49855 0xc2bf. (6570) Charscan searching
for utf-8 character: 50048 0xc380. (6570) Charscan searching for utf-8
character: 50049 0xc381. (6570) Charscan searching for utf-8 character:
50050 0xc382. (6570)
```

If `charscan` is selected but no character list is specified, the `analyze` function performs.

When converting a database's character set, PROUTIL CONVCHAR converts all of the textual data in the database. For more information about character set processing, see *OpenEdge Development: Internationalizing Applications*.

PROUTIL CONVFILE qualifier

Converts a text file from one character set to any other character set.

Syntax

```
proutil -C convfile { file-name convert using table-name
                    | file-name [ analyze ] }
```

Parameters

file-name

Specifies the name of the file you are converting or analyzing.

`convert using` *`table-name`*

Specifies the name of the file containing the conversion table. This file requires the following format:

```
# optional comment lines begin with the # character
SOURCE source-codepage-name
TARGET target-codepage-name
/*000-015*/000  001 002 003 004 005 006 007 008 009 010 011 012 013 014
/*016-031*/016  017 018 019 020 021 022 023 024 025 026 027 028 029 030
/*032-047*/032  033 034 035 036 037 038 039 040 041 042 043 044 045 046
                                        .
                                        .
                                        .
/*240-255*/240  241 242 243 244 245 246 247 248 249 250 251 252 253 254
```

where *`source-codepage-name`* is the name of the character set of *`file-name`*, and *`target-codepage-name`* is the name of the desired character set for *`file-name`*.

The conversion table is composed of 256 cells, numbering from 0 to 255. You can build your own conversion table, or you can use default tables stored in the `prolang` sub-directory of your installation if appropriate. The tables in the `prolang` directory are as follows:

- **`1250-852.dat`** — Converts from code page 1250 to IBM code page 852
- **`1250-il2.dat`** — Converts from code page 1250 to iso8859-2
- **`1254-857.dat`** — Converts from IBM code page 1254 to code page 857
- **`852-1250.dat`** — Converts from IBM code page 852 to code page 1250
- **`852-il2.dat`** — Converts from IBM code page 852 to iso8859-2
- **`857-1254.dat`** — Converts from IBM code page 857 to code page 1254
- **`cn850.dat`** — Converts iso8859-1 to IBM code page 850
- **`cn8859-1.dat`** — Converts IBM code page 850 to iso8859-1
- **`cnnull.dat`** — Performs no conversion; you might want to use this file as a template file for creating new conversion tables
- **`il2-1250.dat`** — Converts from iso8859-2 to code page 1250
- **`il2-852.dat`** — Converts from iso8859-2 to IBM code page 852

If you create your own conversion table, you must understand how PROUTIL CONVFILE uses the table. For each character in *`file-name`*, PROUTIL CONVFILE uses the character's numeric value to index it against the table. When it locates the corresponding cell, PROUTIL CONVFILE replaces the character in the text file with the character value in that cell. Therefore, you must make a copy of the text file before you run the PROUTIL CONVFILE utility, because if the utility does not completely convert the file, the data in it is corrupted.

`analyze`

Displays the number of occurrences of each character in *`file-name`*. You might be able to determine the character set of *`file-name`* by comparing the output of this option against different code pages.

PROUTIL DATAMOVE qualifier

Moves table or sequence data from a multi-tenant source database to a multi-tenant target database for a tenant or group.

Caution: Both the source and target databases must be enabled and configured for multi-tenancy. The source database must be in restricted mode for DATAMOVE to execute. See PROUTIL DBRESTRICT qualifier on page 675 for details on restricting access.

Syntax

```
proutil db-name -C datamove
  [ source source-db-name
     [ [ tenant tenant-name { [table table-name] | [sequence sequence-name] } |
       [ group group-name table table-name]
       [ recs n] ]
  ]
 |
  [ clear all ]
```

Parameters

db-name

Specifies the database receiving the data of the move.

source *source-db-name*

Specifies the source database sending the data of the move.

tenant *tenant-name*

Specifies the tenant receiving the data of the move. If *tenant-name* is not valid, DATAMOVE exits with an error.

group *group-name*

Specifies the group receiving the data of the move. If *group-name* is not valid, DATAMOVE exits with an error.

table *table-name*

Specifies the table receiving the data of the move. If *table-name* is not valid, DATAMOVE exits with an error.

sequence *sequence-name*

Specifies the sequence receiving the data of the move. If *sequence-name* is not valid, DATAMOVE exits with an error.

`recs n`

Specifies the number of records to process in one transaction. If you omit this parameter, DATAMOVE processes 100 records per transaction by default.

`clear all`

Removes all DATAMOVE state information from *db-name*. The `clear all` option is only used if *db-name* needs to be used with a version of OpenEdge earlier than Release 11.3.0 and a DATAMOVE operation terminated prior to completion.

If only the destination (*db-name*) database and source (*source-db-name*) databases are specified, and both databases are in multi-user mode, a menu-driven user interface starts.

The user interface for DATAMOVE allows you to specify multiple tables or sequences to be moved. DATAMOVE moves the data for each table or sequence in parallel by starting a new process for each one.

The user interface main menu is shown in the following figure.

Table 98: PROUTIL DATAMOVE main menu

```
    Progress OpenEdge Tenant/Group Data Move

    Source Database     : source_db
    Destination Database: dest_db

    1. Move Table Data by Tenant
    2. Move Sequence Data by Tenant
    3. Move Table Data by Group
    4. Set Records per Transaction ( Current: 100 )
    5. Quit

Enter your selection:
```

For a description of using the user interface, see Step 3: Moving data with PROUTIL DATAMOVE on page 358

Notes

- Both the restored and target databases must be enabled for multi-tenancy
- The restored and target databases must have identical schema
- The tables instance objects must be allocated
- The tables must have indexes activated
- Dropped object are not supported (only deleted data can be recovered, not table or sequence definitions)
- The restored and target databases must be local (no remote connections)
- You can execute multiple data move commands simultaneously only if both databases are running in multi-user mode

PROUTIL DBANALYS qualifier

Displays statistical information about index, record, and free chain blocks.

Syntax

```
proutil db-name -C dbanalys [ area area-name ]
     [ tenant tenant-name | group group-name | shared ]
     [ -scan ]
     [-csoutput [ -fieldsep sep-value] [ -csfilePrefix pref-name ]
          [ -verbose ] ]
```

Parameters

db-name

Specifies the database you are using.

area *area-name*

Specifies that DBANALYS is to be performed on the *area-name* area only.

tenant *tenant-name*

Specifies that DBANALYS is to be filtered to only provide information related to objects that are owned by *tenant-name*. If *tenant-name* is not valid, DBANALYS exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that DBANALYS is to be filtered to only provide information related to objects that are owned by *group-name*. If *group-name* is not valid, DBANALYS exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

shared

Specifies that DBANALYS is to be filtered to only provide information related to shared objects. Specifying `shared` is only allowed on databases enabled for multi-tenancy.

-scan

Specifies that DBANALYS performs chain analysis with minimal locking (does not turn on table locks).

-csoutput

Specifies that DBANALYS output the analysis data to text files in addition to the screen. The result is that the following files are created:

- *db-name*`.ch.txt` — For chain analysis output
- *db-name*`.ix.txt` — For index analysis output
- *db-name*`.tab.txt` — For table analysis output
- *db-name*`.block.txt` — For block analysis output
- *db-name*`.lob.txt` — For LOB analysis output

For more information about the contents of the files, see the Database analysis output on page 667.

`-fieldsep` *`sep-value`*

When `-csoutput` is specified, *`sep-value`* indicates the value to separate the columns of output. A space is used by default. Accepted values for *`sep-value`* are:

- Any ASCII character between 0x21 and 0x7E
- `sp` — to denote a space (the default)
- `tab` — to denote a the TAB character (`<\t>` or ASCII character 0x09)

`-csfilePrefix` *`pref-name`*

When `-csoutput` is specified, *`pref-name`* indicates a file name prefix. The database name is used by default. All remaining elements of the file name are not customizable.

`-verbose`

When `-csoutput` is specified, the first row of the output files contains header names for each column of output.

Output format

Both non-partitioned and partitioned tables appear in the summary for shared tables in alphabetical order by name. For partitioned tables, the initial partition will appear first, followed by the remainder of the partitions in alphabetical order by partition name.

Example

The following output is a partial sample of the summary section of PROUTIL DBANALYS:

```
                            DATABASE SUMMARY


SUMMARY FOR AREA "Control Area": 1
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                          Records            Indexes            LOBs            Combined
NAME                    Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB._Area             473.0B    0.0     107.0B    0.0       0.0B    0.0     580.0B    0.0
PUB._AreaExtent       981.0B    0.0     129.0B    0.0       0.0B    0.0       1.1K    0.0


SUMMARY FOR AREA "Schema Area": 6
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                          Records            Indexes            LOBs            Combined
NAME                    Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB._Client-Session
                         0.0B    0.0      12.0B    0.0       0.0B    0.0      12.0B    0.0
PUB._Codepage            0.0B    0.0       6.0B    0.0       0.0B    0.0       6.0B    0.0
PUB._Collation           0.0B    0.0       6.0B    0.0       0.0B    0.0       6.0B    0.0
PUB._Constraint          0.0B    0.0      15.0B    0.0       0.0B    0.0      15.0B    0.0
PUB._Constraint-Keys
```

```
                          0.0B    0.0       3.0B    0.0       0.0B    0.0       3.0B    0.0
PUB._Db                   1.1K    0.0      17.0B    0.0       0.0B    0.0       1.2K    0.0
PUB._Db-Detail           36.0B    0.0      37.0B    0.0       0.0B    0.0      73.0B    0.0
PUB._Db-Option          459.0B    0.0     159.0B    0.0       0.0B    0.0     618.0B    0.0
PUB._Field              553.6K    0.5      92.8K    0.1       0.0B    0.0     646.4K    0.6
PUB._Field-Trig           0.0B    0.0       6.0B    0.0       0.0B    0.0       6.0B    0.0
PUB._File                64.9K    0.1       9.2K    0.0       0.0B    0.0      74.1K    0.1
PUB._File-Trig            0.0B    0.0       3.0B    0.0       0.0B    0.0       3.0B    0.0
PUB._Index               35.0K    0.0       8.7K    0.0       0.0B    0.0      43.7K    0.0
PUB._Index-Field         58.1K    0.1       6.9K    0.0       0.0B    0.0      65.0K    0.1
PUB._KeyEvent             0.0B    0.0       9.0B    0.0       0.0B    0.0       9.0B    0.0
PUB._Partition-Policy
                        265.0B    0.0      92.0B    0.0       0.0B    0.0     357.0B    0.0
PUB._Partition-Policy-Detail
                          2.5K    0.0     651.0B    0.0       0.0B    0.0       3.2K    0.0
.
.
.

SUMMARY FOR AREA "tpArea1": 10
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                          Records            Indexes            LOBs            Combined
NAME                    Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.tpCustomer.Initial
                          0.0B    0.0       1.6K    0.0       0.0B    0.0       1.6K    0.0
PUB.tpCustomer.CustomerQtr1
                          0.0B    0.0       1.1K    0.0       0.0B    0.0       1.1K    0.0
PUB.tpInvoice.InvoiceQtr1
                         15.3K    0.0       0.0B    0.0      38.5M   34.6      38.5M   34.6
PUB.tpInvoice.InvoiceQtr5
                          0.0B    0.0       0.0B    0.0       0.0B    0.0       0.0B    0.0
PUB.tpItem                5.0K    0.0       0.0B    0.0       0.0B    0.0       5.0K    0.0
PUB.tpOrder               0.0B    0.0     110.1K    0.1       0.0B    0.0     110.1K    0.1
PUB.tpOrder-Line          0.0B    0.0       2.0K    0.0       0.0B    0.0       2.0K    0.0

SUMMARY FOR AREA "tpArea2": 11
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                          Records            Indexes            LOBs            Combined
NAME                    Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.tpCustomer.CustomerQtr1
                         38.8K    0.0       0.0B    0.0       6.0M    5.4       6.0M    5.4
PUB.tpInvoice.Initial
                          0.0B    0.0       1.1K    0.0       0.0B    0.0       1.1K    0.0
PUB.tpInvoice.InvoiceQtr1
                          0.0B    0.0       1.7K    0.0       0.0B    0.0       1.7K    0.0
PUB.tpItem                0.0B    0.0       4.5K    0.0       0.0B    0.0       4.5K    0.0
PUB.tpLargeField.LrgFldQtr1
                        371.8K    0.3       0.0B    0.0       0.0B    0.0     371.8K    0.3
PUB.tpOrder             244.7K    0.2       0.0B    0.0       0.0B    0.0     244.7K    0.2
PUB.tpOrder-Line         30.3K    0.0       8.1K    0.0       0.0B    0.0      38.5K    0.0


SUMMARY FOR AREA "tpArea3": 12
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                          Records            Indexes            LOBs            Combined
NAME                    Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.tpCustomer.Initial
```

```
                          0.0B    0.0       1.1K    0.0       0.0B    0.0       1.1K    0.0
PUB.tpCustomer.CustomerQtr2
                          0.0B    0.0       0.0B    0.0       0.0B    0.0       0.0B    0.0
PUB.tpCustomer.CustomerQtr5
                          0.0B    0.0       0.0B    0.0       0.0B    0.0       0.0B    0.0
PUB.tpInvoice.Initial
                          0.0B    0.0       1.4K    0.0       0.0B    0.0       1.4K    0.0
PUB.tpInvoice.InvoiceQtr2
                          0.0B    0.0       0.0B    0.0       0.0B    0.0       0.0B    0.0
PUB.tpInvoice.InvoiceQtr5
                          0.0B    0.0       3.0B    0.0       0.0B    0.0       3.0B    0.0
PUB.tpLargeField.LrgFldQtr1
                          0.0B    0.0     286.0B    0.0       0.0B    0.0     286.0B    0.0
PUB.tpLargeField.LrgFldQtr2
                        371.8K    0.3       0.0B    0.0       0.0B    0.0     371.8K    0.3
PUB.tpLargeField.LrgFldQtr5
                          0.0B    0.0       0.0B    0.0       0.0B    0.0       0.0B    0.0


SUMMARY FOR AREA "tpArea4": 13
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                           Records             Indexes              LOBs            Combined
NAME                     Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.BLOBTABLE            31.9K    0.0       0.0B    0.0     391.4K    0.3     423.3K    0.4
PUB.tpCustomer.CustomerQtr2
                          0.0B    0.0       0.0B    0.0       0.0B    0.0       0.0B    0.0
PUB.tpLargeField.LrgFldQtr2
                          0.0B    0.0     286.0B    0.0       0.0B    0.0     286.0B    0.0
PUB.tpLargeField.LrgFldQtr3
                          0.0B    0.0     286.0B    0.0       0.0B    0.0     286.0B    0.0


SUMMARY FOR AREA "tpArea5": 14
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                           Records             Indexes              LOBs            Combined
NAME                     Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.tpCustomer.CustomerQtr5
                          0.0B    0.0       3.0B    0.0       0.0B    0.0       3.0B    0.0
PUB.tpInvoice.Initial
                          0.0B    0.0       1.4K    0.0       0.0B    0.0       1.4K    0.0
PUB.tpInvoice.InvoiceQtr3
                          7.0K    0.0       1.3K    0.0      21.3M   19.2      21.4M   19.2
PUB.tpInvoice.InvoiceQtr5
                          0.0B    0.0       0.0B    0.0       0.0B    0.0       0.0B    0.0


SUMMARY FOR AREA "tpArea6": 15
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                           Records             Indexes              LOBs            Combined
NAME                     Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.tpCustomer.Initial
                          0.0B    0.0     518.0B    0.0       0.0B    0.0     518.0B    0.0
PUB.tpCustomer.CustomerQtr3
                         39.2K    0.0       1.1K    0.0       6.1M    5.5       6.1M    5.5
PUB.tpLargeField.LrgFldQtr3
                        371.8K    0.3       0.0B    0.0       0.0B    0.0     371.8K    0.3
```

```
SUMMARY FOR AREA "tpArea7": 16
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                          Records            Indexes            LOBs            Combined
NAME                    Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.BLOBTABLE            0.0B    0.0       7.0K    0.0       0.0B    0.0       7.0K    0.0
PUB.tpCustomer.CustomerQtr4
                        37.2K    0.0       1.1K    0.0       0.0B    0.0      38.3K    0.0
PUB.tpInvoice.InvoiceQtr4
                        11.4K    0.0       1.3K    0.0       0.0B    0.0      12.7K    0.0
PUB.tpLargeField.LrgFldQtr4
                       371.8K    0.3     286.0B    0.0       0.0B    0.0     372.1K    0.3


SUMMARY FOR AREA "tpArea8": 17
------------------------------------------------------


SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
                          Records            Indexes            LOBs            Combined
NAME                    Size  Tot %       Size  Tot %       Size  Tot %       Size  Tot %
PUB.tpCustomer.CustomerQtr4
                         0.0B    0.0       0.0B    0.0       5.1M    4.6       5.1M    4.6
PUB.tpInvoice.InvoiceQtr4
                         0.0B    0.0       0.0B    0.0      30.9M   27.8      30.9M   27.8

                    ---------------------------------------------------------------------
Total                    2.6M    2.4     278.1K    0.2     108.3M   97.4     111.2M  100.0

Size key:
 B = bytes
 K = kilobytes
 M = megabytes
 G = gigabytes
 T = terabytes


DATABASE BLOCK ANALYSIS:
-----------------------

1 master block(s) found in the database.

8 area block(s) found in the database.

1 control block(s) found in the database.

2 object block(s) found in the database.

80 cluster list block(s) found in the database.

80 cluster allocation block(s) found in the database.

80 object block(s) found in the database.

8 object list block(s) found in the database.

80 object allocation block(s) found in the database.

7089 free block(s) found in the database.

1 sequence block(s) found in the database.

214 empty block(s) found in the database.
```

```
26630 total blocks found in the database.

Database analysis complete Wed Aug 12 14:16:50 2015
```

Output format

In a complete report where no optional elements are specified on the command line, shared objects (both partitioned and non-partitioned) appear in alphabetical order by name. Partitions of partitioned tables are ordered as follows: the initial partition, then the remainder of the partitions in alphabetical order by partition name. When partitioned indexes are listed, the indexes appear in alphabetical order. For each partitioned index, the Index of Indexes appears first, followed by the remaining partitions in alphabetical order by partition name.

The chain analysis report appears in internal storage order (as stored in the object block for the area) with a column to identify **Partition/Tenant/Group**. The format of this column is as follows: *type*:*name*:*id* where *type* is **P** for Partition, **T** for Tenant, or **G** for Group; *name* is the partition name, tenant name, or group name; and *id* is the partition-id, tenant-id, or group-id. The column is blank if there is no partition, tenant, or group.

The output line listing the number of blocks on chains associated with the Index of Indexes displays "_Partition-Policy-Detail._Partition-Internal-Value" in the Object column, and P:*FullyQualifiedOwningTableName*:*TableId* in the **Partition/Tenant/Group** column. Composite partitions display C:*partition-name*:*partition-Id*.

Note

- The output of PROUTIL DBANALYS is the combination of the three individual analysis utilities (CHANALYS, IDXANALYS, and TABANALYS) combined.
- When DBANALYS performs chain analysis, only the length of the RM chain is reported. You must run CHANALYS to get a more detailed report including the list of free chain blocks.
- If you specify a specific area for analysis, and the area contains LOB data, but not the corresponding records, a warning message is issued indicating that totals cannot be calculated for the LOBs in the LOB Summary and Database Summary sections of the output.

Database analysis output

The output of the database analysis tools, DBANALYS, IDXANALYS, CHANALYS, and TABANALYS, can be exported to a set of text files, in addition to the tabular reports output to the screen. These files are easily importable to an OpenEdge database or spreadsheet for analysis and trending. Both the naming of the text files, and the layout of the contents is configurable by command line parameters.

To create the text files, add the parameter `-csoutput` to the command line, as shown:

```
proutil db-name -C dbanalys -csoutput
```

The result is that the following files are created:

- *dbname* `.ch.txt` — For chain analysis output
- *dbname* `.ix.txt` — For index analysis output
- *dbname*`.tab.txt` — For table analysis output
- *dbname* `.block.txt` — For block analysis output

- *dbname* .lob.txt — For LOB analysis output

DBANALYS produces all five text files; the remaining tools produce the text file(s) specific to the command plus the block file.

If you restrict the analysis to a specific area with the use of the area parameter on the PROUTIL command line, the file names are altered to include the area number before the final ".txt" extension, *dbname*.block*area#*.txt. Any existing file is overwritten.

You can substitute the *dbname* prefix of the file name by specifying the -csfilePrefix parameter, and supply a different value. For example, if you wanted to add a date to the file name prefix, you could enter:

```
proutil proddb -C dbanalys -csoutput -csfilePrefix proddb_sep01
```

and the following files are created:

- proddb_sep01.ch.txt
- proddb_sep01.ix.txt
- proddb_sep01.tab.txt
- proddb_sep01.block.txt
- proddb_sep01.lob.txt

Output file format

By default, the text files contain lines of output, where each column is delimited by a space (ASCII character 0x20). Character values are quoted ("") and numeric values represent the number of bytes and are raw numbers. You can modify the separator by specifying an alternative with the -fieldsep parameter, as shown:

```
proutil db-name -C dbanalys -csoutput -fieldsep sep-value
```

Acceptable values for *sep-value* are:

- Any ASCII character between 0x21 and 0x7E
- sp — to denote a space (the default)
- tab — to denote a the TAB character (<\t> or ASCII character 0x09)

By default, the text files contain only data. If you want to add column headings, you can add the parameter -verbose to the command line, as shown:

```
proutil db-name -C dbanalys -csoutput -verbose
```

The resulting output file contains column headings in the first line.

The contents of the files are described in the following table.

Table 99: Text file contents for -csoutput

File type	Column header
Chain (*dbname*.ch.txt)	Time Stamp
	DBUID
	DB name
	Area
	Area #
	Partition Type
	Tenancy
	Tenancy #
	Partition
	Partition #
	Object type
	Owner
	Object
	Object #
	Chain Type
	Num Blocks

File type	Column header
Index (*dbname*.ix.txt)	Time Stamp
	DBUID
	DB name
	Area
	Area #
	Partition Type
	Tenancy
	Tenancy #
	Partition
	Partition #
	Owner
	Table
	Table #
	Index
	Index #
	Fields
	Levels
	Blocks
	Size
	Utilization
	Factor

File type	Column header
Table (*dbname*.tab.txt)	Time Stamp
	DBUID
	DB name
	Area
	Area #
	Partition Type
	Tenancy
	Tenancy #
	Partition
	Partition #
	Owner
	Table
	Table #
	Records
	Size
	Min
	Max
	Mean
	Frag Count
	Frag Factor
	Scatter Factor

File type	Column header
LOB (*dbname*.lob.txt)	Time Stamp
	DBUID
	DB name
	Area
	Area #
	Partition Type
	Tenancy
	Tenancy #
	Partition Policy
	Partition
	Partition #
	Owner
	Table
	LOB Obj #
	LOBs
	Size
	Min
	Max
	Mean
Block (*dbname*.block.txt)	Time Stamp
	DBUID
	DB name
	Area
	Area #
	Block Type
	Num Blocks

Additional information to note about the contents of the text files:

- Totals are not reported
- The time stamp of each row is identical, indicating that each row came from a single run of the analysis
- The DBUID is provided to identify rows if the output from multiple sources is loaded into a trending database
- The CHANALYS output does NOT contain the dbkey information, it only include the chain analysis summary information as in the dbanalys report.

PROUTIL DBAUTHKEY qualifier

Sets an authorization key for a database.

Syntax

```
proutil db-name -C dbauthkey old-key new-key
```

Parameters

db-name

Specifies the database you are using.

old-key

Specifies the old authorization key.

new-key

Specifies the new authorization key.

Notes

- You use the DBAUTHKEY qualifier to set, change, and remove authorization keys. When you compile source code, the value of this key is included in your r-code. CRC-based r-code that does not include the correct authorization key will not run against your database. The following table lists the values you must enter for each task:

Task	old-key value	new-key value
Set the authorization key	Plus sign (+)	Authorization key
Change the authorization key	Current authorization key	New authorization key
Remove the authorization key	Current authorization key	Plus sign (+)

- Once you set the authorization key, do not forget it. You cannot change or remove the authorization key without knowing its current value.
- After you set, change, or remove the authorization key for a database, you can use the RCODEKEY qualifier to update your r-code without recompiling it.

PROUTIL DBIPCS qualifier

Displays status information for shared-memory segments attached by all OpenEdge databases on the system. This qualifier is valid only on UNIX.

Syntax

```
proutil -C dbipcs
```

Example

The following output is a sample display of PROUTIL DBIPCS:

```
PROGRESS SHARED MEMORY STATUS
 ID  ShMemVer  Seg#  InUse   Database
 68     -       -     -      (not PROGRESS)
100     3       0     Yes    /db/work5/sports
101     3       1     -      /db/work5/sports
120     3       0     No     /db/work5/test
```

`ID`

Indicates the shared-memory ID.

`ShMemVer`

Specifies the shared-memory version.

`Seg#`

Indicates the shared-memory segment number. One database can own more than one segment.

`InUse`

Specifies whether the segment is in use. Yes or No values are displayed only if the segment is number zero (0). All other segments show a hyphen (-). To determine whether a set of segments is in use, check the **InUse** value of segment 0 for the relevant database.

`Database`

Represents the full pathname of the database.

Notes

- The PROMON Activity screen shows the amount of shared memory used by OpenEdge processes and the number of shared-memory segments.
- On UNIX, use `ipcs -m` to determine the size and number of shared-memory segments in use. Subtract from `SHMALL` to see how many shared-memory segments are left. If the broker dies or is killed by means other than the database shutdown command, it does not release attached shared-memory segments. Use PROUTIL DBIPCS to verify shared memory is frozen; use the UNIX command `ipcrm -m` to free it.

PROUTIL DBRESTRICT qualifier

Enables or disables write access to the database for processes other than the specified utility.

Syntax

```
proutil dbname -C dbrestrict restrict-type restrict-action
```

Parameters

db-name

Specifies the database where you are controlling access.

restrict-type

Specifies the operation to allow (restricting other access). Accepted values are:

- `datamove` — restrict all operations, excepting PROUTIL DATAMOVE
- `partitioncopy` — restrict all operations, excepting PROUTIL PARTITIONMANAGE COPY
- `rollforward` — restrict all operations, excepting RFUTIL ROLL FORWARD

restrict-action

Specifies starting, ending, or querying the status of the restriction. Accepted values are:

- `enable` — start the restriction
- `disable` — end the restriction
- `status` — query the restriction, returning a message indicating if the restriction is enabled or disabled

Use DBRESTRICT to protect your database from modification by other processes during the execution of the identified utility. When enabled, access is allowed as follows:

- For DATAMOVE, the following is allowed:
  - PROUTIL DATAMOVE
  - Database brokers
  - PROMON
  - PROSHUT
  - PROUTIL DBRESTRICT (to modify or disable the restriction)
- For PARTITIONCOPY, the following is allowed:
  - PROUTIL PARTITIONMANAGE COPY
  - PROUTIL PARTITIONMANAGE VIEW
  - Database brokers
  - PROMON

- PROSHUT
- PROUTIL DBRESTRICT (to modify or disable the restriction)

- For ROLL FORWARD, the following is allowed:
  - RFUTIL ROLL FORWARD
  - PROSTRCT ADD
  - PROSTRCT LIST
  - PROUTIL IDXCHECK
  - PROUTIL IDXANALYS
  - PROUTIL TABANALYS
  - PROUTIL DBANALYS
  - PROUTIL DBRESTRICT (to modify or disable the restriction)

Example

- **Datamove restriction**

  The DATAMOVE utility requires restricted access to the database. The following example shows how to enable restricted access on a database named `mydb`:

```
proutil mydb -C dbrestrict datamove enable
```

  Once you have completed the data move, remove the access restriction as shown:

```
proutil mydb -C dbrestrict datamove disable
```

- **Migrating a restriction from rollforward to datamove**

  If you first restrict access to your database for roll forward, you can migrate the restriction to data move without first disabling the roll forward restriction.

  1. Verify that the database is in rollforward restricted mode, as shown:

```
proutil mydb -C dbrestrict rollforward status
```

  2. The following status is returned:

```
Restricted Access Mode is enabled for RollForward (17073)
```

  3. Migrate the restriction:

```
proutil mydb -C dbrestrict datamove enable
```

4. The following status is returned:

```
Database Restricted Access mode has been disabled for RollFoward (17039)
Database Restrict Access has been enabled for DataMove (17037)
```

Notes

- You must truncate the BI of your database prior to enabling DBRESTRICT.
- Your database must be offline when you run DBRESTRICT.
- Your database restriction status is shown in the PROMON Database Status displays.
- The Restricted Modes status is available in the read only `_dbStatus` VST, in the field named `_DbStatus-RestrictMode`. The field label is Restricted Access Mode, and supported field values are: DataMove, Rollforward, and disabled.
- If you run DBRESTRICT on a Replication source database, you must still supply all your replication parameters to start a broker, however, the replication server will not start until you have disabled the restriction.

PROUTIL DEALLOCATE qualifier

Deallocate a multi-tenant table from a database enabled for multi-tenancy.

Syntax

```
proutil dbname -C deallocate [owner.] table-name
          tenant [tenant-name | group group-name ]
```

Parameters

db-name

Specifies the database with the table to be deallocated.

[*owner.* **]** *table-name*

Specifies the name of the multi-tenant table to be deallocated. *owner* is required if the table owner is not PUB.

tenant *tenant-name*

Specifies the tenant whose table partition is to be deallocated. *tenant-name* cannot be a member of a group.

group *group-name*

Specifies the group whose table partition is to be deallocated.

Notes

- DEALLOCATE works both online and offline.
- DEALLOCATE only executes on multi-tenant tables in a multi-tenancy enabled database.
- Multiple instances of DEALLOCATE can run online at the same time, provided they are not attempting to deallocate the same table for the same tenant or group.
- The super-tenant can deallocate other tenant's or group's table instance.
- If DEALLOCATE detects any users (particularly no-lock readers) when the deallocation is executing, the utility waits for them to exit from the transition state printing the following messages:

```
Waiting for user(s) to finish record operation.
If you'd like to cancel this operation, press CTRL-C.
```

PROUTIL DESCRIBE qualifier

Displays descriptive information about the database and enabled features.

Syntax

```
proutil dbname -C describe
```

Parameters

db-name

Specifies the database to describe.

PROUTIL DESCRIBE provides a basic description of the database and details which features are enabled for the database.

The following figure provides a sample of the PROUTIL DESCRIBE output.

Figure 56: PROUTIL DESCRIBE sample output

```
OpenEdge Database Description

Database Name               : /usr1/dbs/sample_db
Version                     : 173.0
Block Size                  : 8192
Largest Cluster             : 64
Create Date                 : Mon Dec 10 13:04:58 2012
Last Open Date              : Mon Dec 10 13:25:24 2012
Prior Open Date             : Mon Dec 10 13:25:24 2012
Schema Change Date          : Mon Dec 10 13:05:09 2012

 Before Imaging information
  Block Size                : 8192
  Cluster Size (16K Units)  : 32
  Last Open Date            : Mon Dec 10 13:05:25 2012
  Bi Encryption             : Enabled
```

```
After Imaging Information
 Block Size                : 8192
 Begin Date                : Mon Dec 10 13:05:25 2012
 Last AIMAGE NEW           : Mon Dec 10 13:05:26 2012
 Currently Busy Area       : 13
 Current File Number       : 2
 Ai Encryption             : Enabled

Backup Information
 Last Full Backup Date     : Mon Dec 10 13:05:17 2012
 Last Incremental Backup   : *** Not yet performed ***

Database Features

   ID   Feature                            Active  Details
 ----  --------------------------------  ------  -------
    1  OpenEdge Replication              Yes     Source database
    7  JTA                               Yes
    9  64 Bit DBKEYS                     Yes
   10  Large Keys                        Yes
   11  64 Bit Sequences                  Yes
   13  Encryption                        Yes
   14  Multi-tenancy                     Yes
   15  Concurrent JTA and Replication    Yes
```

The PROUTIL DESCRIBE output contains the following fields:

`Database Name`

The fully qualified name of the database.

`Version`

The internal database version number.

`Block Size`

The block size of the database, measured in bytes.

`Largest Cluster`

The largest cluster size, in blocks, for all of the database data areas.

`Create Date`

The date and time the database was created.

`Last Open Date`

The date and time the database was last opened.

`Prior Open Date`

The date and time of the penultimate database open.

`Schema Change Date`

The date and time of the last schema change.

`Before Imaging Information`

The details of the database Before-Image areas, including:

- `Block Size` — The size (in bytes) of the before-image blocks
- `Cluster Size (16K Units)` — The size of the before-image cluster, measured in units of 16K
- `Last OpenDate` — The date and time the before-image log was last opened
- `Bi Encryption` — Indicates if BI encryption is enabled or disabled

`After Imaging Information`

The details of After-Imaging areas, if after-imaging is enabled, including:

- `Block Size` — The size (in bytes) of the after-image blocks
- `Begin Date` — The date and time after-imaging was started
- `Last AIMAGE NEW` — The date and time the last RFUTIL AIMAGE NEW command was executed, forcing an extent switch
- `Currently Busy Area` — The area number of the currently busy after-image area
- `Current File Number` — The current after-image file number
- `Ai Encryption` — Indicates if AI encryption is enabled or disabled

`Two-Phase Commit Information`

The details of two-phase commit, if enabled, including:

- `Nickname` — The 2PC database nickname
- `Coordinator Priority` — The 2PC coordinator priority for the database

`Backup Information`

The details of recent database backup activity, including:

- `Last Full Backup Date` — The date and time of the last full backup
- `Last Incremental Backup` — The date and time of the last incremental backup

`Database Features`

The details of the enabled features of the database, including:

- `ID` — The ID number of the feature.
- `Feature` — The name of the feature.
- `Active` — Is the feature active? Yes/No. Features, such as Auditing, can be enabled, but not active.
- `Details` — Provides special information pertinent to the feature. For example, for OpenEdge Replication, the database type is listed.

The table below lists the possible `ID` and `Feature` names.

Table 100: Database features

ID	Feature
1	OpenEdge Replication
2	Failover Clusters
3	DataXtend Remote Edition
5	Large Files
6	Database Auditing
7	JTA
8	After Image Management/Archiver
9	64-bit DB Keys
10	Large Keys (for indexes)
11	64-bit Sequences
12	DataXtend Integration Edition
13	Transparent Data Encryption
14	Multi-tenancy
19	Roll Forward Restricted Mode
21	Table Partitioning
26	Authentication Gateway
27	Change Data Capture

Notes

DESCRIBE can be run against a database that is online or offline.

PROUTIL DISABLEAUDITING qualifier

Disables auditing for a database.

Syntax

```
proutil db-name -C disableauditing [-userid username[-password passwd] ]
```

Parameters

db-name

Name of the database to have auditing disabled.

-userid *username*

Identifies a user with the Audit Administrator privilege required to run this utility.

-password *passwd*

Password for the privileged user.

Ensuring the security of your audit data requires that the audit data tables be **empty** when you disable auditing. If a database has data in the audit data tables, _aud-audit-data and _aud-audit-data-value, when the command is issued, the database is **deactivated** rather than disabled.

The following message is displayed when auditing is deactivated:

```
Auditing was not fully disabled because auditing data tables are not empty.
(13647)

Auditing has been deactivated, no additional auditing records will be
recorded. (13649)
```

When the audit data tables are empty, and auditing is successfully **disabled**, the following message is displayed:

```
Auditing has been disabled for database db-name. (12490)
```

For more information on auditing states, see Auditing states on page 182.

Notes

- Only users with the Audit Administrator privilege are allowed to run this utility.
- The database must be offline when disabling auditing.
- Disabling auditing does not remove any recorded audit data or the auditing tables.
- Access to the audit data remains restricted to authorized users when auditing is disabled.

PROUTIL DISABLEAUTHGATEWAY qualifier

Disables the OpenEdge Authentication Gateway for a database.

Syntax

```
proutil db-name -C disableauthgateway
```

Parameters

db-name

Name of the database to have use of the OpenEdge Authentication Gateway disabled.

DISABLEAUTHGATEWAY removes the `_db-option` record with `_db-option-code="_db.sts.url"` and `_db-option-type=3`. (These are the values set by using the STS URL Utility prior to enabling the database to use the Authentication Gateway.) After disabling use of the OpenEdge Authentication Gateway, connection role enforcement no longer takes place. If the database is enabled for connection role enforcement at the time DISABLEAUTHGATEWAY executes, the following warning message is reported:

```
Warning: Disabling the OpenEdge Authentication Gateway feature disables
enforcement of connection role security.
```

After successfully executing DISABLEAUTHGATEWAY, the following message is reported:

```
The Authentication Gateway has been disabled for database dbname.
```

If auditing is enabled on your database, Audit event 11902 is recorded.

PROUTIL DISABLEB2 qualifier

Removes an area from alternate buffer pool processing.

Syntax

```
proutil db-name -C disableB2 area-name
```

Parameters

db-name

Name of the database where you are removing an area from the alternate buffer pool.

area-name

Name of the area (Type I or Type II) you are removing from the alternate buffer pool.

The Alternate buffer pool is a pool of buffers maintained independently from the primary buffer pool. Disabling an area from the alternate buffer pool indicates that it will consume buffers from the primary buffer pool.

Notes

- DISABLEB2 must be run offline.
- DISABLEB2, and alternate buffer pool support in general, are only supported with an Enterprise database license.

PROUTIL DISABLECDC qualifier

Disables Change Data Capture (CDC) from your database.

Syntax

```
proutil db-name -C disablecdc
```

Parameters

db-name

The name of the database where you are disabling CDC.

Successful execution of DISABLECDC removes the CDC schema, the change tracking table, and the CDC feature bit from your database.

Prior to disabling CDC, all CDC policies and change tables must be deleted.

Disabling CDC performs the following:

- Checks that the _CDC-Table-Policy and _CDC-Field-Policy tables are empty, which signals that all Change Tables have been dropped from the database. If any tables are found, DISABLECDC exits with an error.
- Removes the Table Policy table and its indexes.
- Removes the Field Policy table and its indexes.
- Removes Change Tracking Table and its indexes. If the table has records in it, removing the table effectively deletes all its records.
- Deactivates the Feature Bit Map for CDC.
- If Auditing is enabled, writes an Audit Event record for disabling CDC.

See:

- PROUTIL ENABLECDC qualifier on page 698

PROUTIL DISABLEENCRYPTION qualifier

Disables transparent data encryption on your database.

Syntax

```
proutil db-name -C disableencryption
       [-Passphrase] [ [-userid userid] [-password password] ]
```

Parameters

db-name

Name of the database where you are disabling encryption.

-Passphrase

Specifies that the user must be prompted for a passphrase to authenticate the key store, before running this command.

`-userid` *`userid`* `-password` *`password`*

Specifies the userid and password of an authenticated database administrator.

Successful execution of DISABLEENCRYPTION has the following impact on your database:

- BI encryption is disabled
- AI encryption is disabled
- All data is decrypted
- All encryption policies are removed
- The key store is archived (renamed *`dbname`*`.ks.bk`)

Notes

- You cannot disable BI encryption while your database is online. You can run DISABLEENCRYPTION for an online database, but BI encryption will not be disabled until the BI is truncated and the database is restarted. DISABLEENCRYPTION issues a warning.
- DISABLEENCRYPTION is interruptible, but will not restore the database to the state it was in prior to starting the disabling processing. If you interrupt the DISABLEENCRYPTION command, your database might be at one of the following points:
  - BI encryption is disabled (if run offline)
  - AI encryption is disabled
  - All ciphers of current policies are set to NULL
  - Add encrypted data are decrypted
  - All encryption policies are removed
  - Encryption is fully disabled
- Because processing DISABLEENCRYPTION can take a long time, you may want to decrypt your data before issuing this command. Decrypt your data by setting your encryption policies to the NULL cipher and updating your data with PROUTIL EPOLICY.

PROUTIL DISABLEJTA qualifier

Disables database participation in JTA transactions.

Syntax

```
proutil db-name -C disablejta
```

Parameters

`db-name`

Name of the database to disable from supporting JTA transactions.

Notes

For more information on JTA transactions, see Distributed Transaction Processing on page 255

PROUTIL DISABLEKEYEVENTS qualifier

Disables the storage of key events for the database.

Syntax

```
proutil db-name -C disablekeyevents
```

Parameters

db-name

Name of the database being disabled for key event storage.

Notes

- The database may be online when disabling key events.
- Events in the `_KeyEvt` table are not deleted when PROUTIL DISABLEKEYEVENTS is run.

PROUTIL DISABLEMULTITENANCY qualifier

Disables the support of multi-tenant tables for the database.

Syntax

```
proutil db-name -C disablemultitenancy
```

Parameters

db-name

Name of the database were multi-tenant tables are being disabled.

Notes

- The database must be offline when disabling multi-tenancy.
- The database must have previously been enabled for multi-tenancy to successfully disable.
- Before DISABLEMULTITENANCY can execute successfully:
  - All tenants and groups must be deleted from the database

PROUTIL DISABLEREADONLYPARTITIONS qualifier

Disables the ability to change a partition to a read-only state in a table-partitioned database.

Syntax

```
proutil db-name -C disablereadonlypartitions
```

Parameters

db-name

Name of the database to be disabled for read-only partitions.

The database must be enabled for read-only partitions and must be offline to disable read-only partitions. You must change any read-only partitions to read-write prior to executing DISABLEREADONLYPARTITIONS.

See PROUTIL PARTITIONMANAGE ALTER qualifier on page 746 for information on changing partition states between read-only and read-write.

PROUTIL DISABLETABLEPARTITIONING qualifier

Disables table partitioning from your database.

Syntax

```
proutil db-name -C disabletablepartitioning
```

Parameters

db-name

The name of the database where you are disabling table partitioning.

Successful execution of DISABLETABLEPARTITIONING removes the ability to partition tables from your database.

Prior to disabling table partitioning:

- All existing table partitions must be removed from your database
- Read-only partitions, if enabled, must be disabled
- Single-partition index rebuild, if enabled, must be disabled

See:

- PROUTIL ENABLETABLEPARTITIONING qualifier on page 704
- PROUTIL DISABLEREADONLYPARTITIONS qualifier on page 687

PROUTIL DISPTOSSCREATELIMITS qualifier

Displays toss and create limits for the specified area.

Syntax

```
proutil db-name -C disptosscreatelimits area-name
    [ tenant tenant-name | group group-name | shared ]
```

Parameters

db-name

Name of the database to have toss and create limits displayed.

area-name

The area to have toss and create limits displayed. You can specify an area name or area number.

`tenant` *tenant-name*

Specifies that only the area limits associated with *tenant-name* are to be displayed. If *tenant-name* is not valid, is part of a group, or is not associated with a partition in the area, DISPTOSSCREATELIMITS exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy and Type II areas.

`group` *group-name*

Specifies that only the area limits associated with *group-name* are to be displayed. If *group-name* is not valid or is not associated with partition in the area, DISPTOSSCREATELIMITS exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy and Type II areas.

`shared`

Specifies that only the area limits on shared objects in the area is to be displayed. Specifying `shared` is only allowed on databases enabled for multi-tenancy and Type II areas.

Notes

For databases enabled for multi-tenancy, if `tenant`, `group`, or `shared`, is not specified, and the area has multi-tenant objects, the limits are displayed for all objects in the area.

PROUTIL DUMP qualifier

Performs a binary dump of a database table to a file.

```
proutil db-name -C dump [owner-name.] table-name directory
      [ tenant tenant-name | group group-name | partition partition-name
          | composite initial ] [ -index num ]
      [ [ -thread n ] [ -threadnum nthreads ] ]
      [ -dumplist dumpfile ]
      [ -Cipher cipher-id ]
```

Parameters

db-name

Specifies the database where the dump will occur. If the database is not within the current working directory, you need to define the complete path.

owner-name

Specifies the owner of the table containing the data you want to dump. You must specify an owner name unless the table's name is unique within the database, or the table is owned by PUB. By default, ABL tables are owned by PUB.

table-name

Specifies the name of the table containing the data you want to dump.

directory

Specifies the name of the target directory where the data will be dumped.

tenant *tenant-name*

Specifies that the DUMP operation is to be performed **only** on the table partitions of *tenant-name*. If *tenant-name* is not a valid tenant, DUMP exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the DUMP operation is to be performed **only** on the table partitions of *group-name*. If *group-name* is not a valid group, DUMP exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

partition *partition-name*

Specifies that the DUMP operation is to be performed **only** on the table partition *partition-name*. If *partition-name* is not valid, DUMP exits with an error. Specifying `partition` is only allowed on databases enabled for table partitioning.

`composite initial`

Specifies that the DUMP operation is to be performed on the composite initial partition. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

`-index` *num*

Specifies the index to use to dump the table's contents. Note that word indexes are not allowed. If you choose not to use this option, the command uses the primary index to dump the table.

`-thread` *n*

For databases with an Enterprise license, indicate if an online dump is threaded. Specify zero (`0`) for a single-threaded dump, one (`1`) for a threaded dump.

`-threadnum` *nthreads*

For a threaded dump, specify the maximum number of threads to create. The default value is the number of system CPUs. The actual number of threads created may be less than *nthreads*. PROUTIL DUMP determines the number of threads to create based on the complexity of the index the DUMP follows.

`-dumplist` *dumpfile*

Create a file, *dumpfile*, that lists all the files created by this utility. The file, *dumpfile*, can be used as an input parameter to binary load (PROUTIL LOAD).

`-Cipher` *cipher-id*

For an Enterprise database enabled for Transparent Data Encryption, specify either `-Cipher 11` or `-Cipher 12` to encrypt the output of the PROUTIL DUMP with a password-based encryption cipher (PBE). You are prompted to enter the passphrase for the cipher. You must remember this passphrase; you will be prompted to enter the passphrase when loading the binary dump.

PROUTIL DUMP writes data from a table to a dump file or files. When the procedure finishes, it reports the number of records written to the dump file. For a threaded dump, the number of records written by each thread is also reported.

PROUTIL DUMP selects some or all of the records in the specified table to dump, based on the parameters:

- When dumping the contents of a multi-tenant database, if no tenant or group is specified, then the entire table is dumped.
- When dumping the contents of a partitioned table, if no partition is specified, then the entire table is dumped.

Note: You can refine the rows dumped from a table based on data values using PROUTIL DUMPSPECIFIED qualifier on page 692.

PROUTIL DUMP output

PROUTIL DUMP data from a table to a file or multiple files. You can easily save the names of the created files by specifying a list file with the `-dumplist` qualifier. The names of the files created during the dump process follow a number of conventions depending on what is being dumped:

- The name of the dump files depends on the owner of the table. By default, ABL tables are owned by PUB. When tables owned by PUB are dumped to a file by an offline or single-threaded online dump, the filename is the table name with `.bd` appended. For example, *tablename*`.bd`.

However, when tables owned by anyone other than PUB are dumped to a file, the resulting filename contains the owner name and table name. For example, *ownername_tablename*.bd

- On systems that have a 2GB-file-size limitation, PROUTIL DUMP creates multiple files when you dump a table larger than 2GB. For example, when you dump data from a table with the name "customer" that is 6.4GB, PROUTIL DUMP creates four binary dump files: customer.bd, customer.bd2, and customer.bd3, each of which is approximately 2GB, and customer.bd4, which is approximately 0.4GB. The PROUTIL DUMP procedure adds header blocks to the binary dump files. As a result, the total size of the binary dump files is slightly larger than the table itself.

  On systems without 2GB-file-size limitation, PROUTIL DUMP, running singly threaded, creates only one binary dump file regardless of the size of the table.

- For all threaded dumps, PROUTIL DUMP creates multiple files, one for each thread. The first thread creates the file *tablename*.bd; the second thread creates the file *tablename*.bd2; each additional thread creates a file with an incremental number, *tablename*.bd*n*.

- When dumping the contents of a multi-tenant or table-partitioned database, and a tenant, group, or partition is specified, the naming convention of the dump file is modified as shown:

```
table-name_ID{T|G|P|C}_TPGCName.bdn
```

  The elements of the file name are described in the following table.

Table 101: Binary dump file name description

File name element	Description
table-name	The name of the table being dumped. If the table is not owned by PUB, then the owner is prepended to the table name.
ID	The tenant, group (without negative sign), or partition ID.
{T|G|P|C}	Identifier: • T indicates the *ID* is a tenant ID. • G indicates the *ID* is a group ID. • P indicates the *ID* is a table partition ID. • C indicates the *ID* is the composite partition ID.
TPGCName	The name of the tenant, group, table partition, or composite initial partition that corresponds to *ID*.
n	The number of the binary dump file when more than one file is created during the dump of the table. The value of *n* is determined either by the size of the file or the number of threads created.

- For threaded dumps of a partitioned table, if you specify a local index with -index, each partition is dumped sequentially. The number of threads created for each partition (and therefore the number of output files) depends on the depth of the index tree for the partition. If the index has only one level for a given partition, the dump of that partition is not threaded. The number of threads created for each partition is output to the screen as the dump progresses.

If a global index is specified to dump a partitioned table, then the dump is not broken out by partition.

For example, if you dump the list-range partitioned table, `history`, by a local index, the dump files created are named by partition: `history_1P_hist_lr_1.bd`, `history_1P_hist_lr_1.bd2`, etc. If the same table is dumped by a global index, the dump files created are named by the table: `history.bd`, `history.bd2`, etc.

Notes

- See Dumping and Loading on page 411 for more information about the DUMP qualifier.
- The PROUTIL DUMP and LOAD utilities use cyclic redundancy check (CRC) values to establish the criteria for loading.

  The OpenEdge database provides a flexible storage architecture and the ability to relocate objects, such as tables and indexes, while the database remains online. As a result, when you perform a binary load operation, the table numbers in a binary dump file might not match the table numbers in the target database. Therefore, when you perform a binary load operation, the criteria for loading tables is based solely on cyclic redundancy check (CRC) values, and not table numbers.

  For example, when you dump a table, the PROUTIL utility calculates a CRC value for the table and stores it in the header of the binary dump file. When you load the table, PROUTIL matches the CRC value stored in the header with the CRC value of the target table. The values must match or the load is rejected.

  You can load a binary dump file created with a previous version of the PROUTIL DUMP utility, because the current version of PROUTIL LOAD uses the CRC value established when the file was originally dumped. Consequently, the database maintains backwards compatibility.
- If the file specified by the `-dumpfile` option exists, PROUTIL DUMP overwrites the existing file.
- If you choose a global index to dump a specific table partition, DUMP must perform a table scan, and performance will be slower than if a local index is selected.
- You cannot specify the composite initial partition with the `partition` qualifier, use `composite initial`.
- PROUTIL DUMP supports dumping binary large objects (BLOBS).
- Dumped data can only be loaded to a database with the same code page as the source database. Use the ASCII dump and load utilities to perform a code page conversion with your dump and load. For more information on the ASCII utilities, see Dumping user table contents with a Data tool on page 424.
- You cannot use DUMP to dump protected audit data. For more information on auditing and utility security and restrictions, see Auditing impact on database utilities on page 193.
- When dumping the contents of a table encrypted with Transparent Data Encryption, the output of the binary dump is **not** encrypted by default. Encrypt the dump file (container security) by specifying either `-Cipher 11` or `-Cipher 12` and entering a passphrase for the PBE cipher. You are prompted for the passphrase when loading the contents of the dump file.

PROUTIL DUMPSPECIFIED qualifier

Performs a binary dump of selected records.

Syntax

```
proutil db-name -C dumpspecified [owner-name.] table-name.field-name
          operator1 low-value [ AND operator2 high-value]
             directory
          [ tenant tenant-name | group group-name |
                    partition partition-name | composite initial]
          [-preferidx index-name]
          [ -Cipher  cipher-id]
```

Parameters

db-name

Specifies the database where the data being dumped resides. You must completely define the path.

owner-name

Specifies the owner of the table containing the data you want to dump. You must specify an owner name unless the table's name is unique within the database, or the table is owned by PUB. By default, ABL tables are owned by PUB.

table-name

Specifies the name of the table containing the data you want to dump.

field-name

Specifies the name of the index field, or the name of a component field of a compound index, to be used to select the data you want to dump. If *field-name* is an index or the name of a component field of a compound index, it is used to select the data you want to dump, otherwise the primary index is used.

operator1

Specifies the first operator in the range: EQ (equals to), GT (greater than), LT (less than), GE (greater than or equal to), or LE (less than or equal to).

low-value

Specifies the first value against which the *field-name* value will be compared.

AND

Specifies a bracketed range for the binary dump. The first value and operator, combined with the second value and operator, define a subset. You cannot use AND to dump two sets of non-contiguous result.

operator2

Specifies the second operator in the range: LT (less than), or LE (less than or equal to).

high-value

Specifies the first value against which the *field-name* value will be compared.

directory

Specifies the name of the target directory where the data will be dumped. You must specify a directory.

tenant *tenant-name*

Specifies that the DUMPSPECIFIED operation is to be performed **only** on the table partitions of *tenant-name*. If *tenant-name* is not a valid tenant, DUMPSPECIFIED exits with an error. Specifying tenant is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the DUMPSPECIFIED operation is to be performed **only** on the table partitions of *group-name*. If *group-name* is not a valid group, DUMPSPECIFIED exits with an error. Specifying group is only allowed on databases enabled for multi-tenancy.

partition *partition-name*

Specifies that the DUMPSPECIFIED operation is to be performed **only** on the table partition *partition-name*. If *partition-name* is not valid, DUMPSPECIFIED exits with an error. Specifying partition is only allowed on databases enabled for table partitioning.

composite initial

Specifies that the DUMP operation is to be performed on the composite initial partition. Specifying composite initial is only allowed on databases enabled for table partitioning.

-preferidx *index-name*

By default, DUMPSPECIFIED uses the primary index to sort rows. Specifying a different index with -preferidx orders the rows by the specified index.

-Cipher *cipher-id*

For an Enterprise database enabled for Transparent Data Encryption, specify either -Cipher 11 or -Cipher 12 to encrypt the output of the PROUTIL DUMPSPECIFIED with a password-based encryption cipher (PBE). You are prompted to enter the passphrase for the cipher. You must remember this passphrase; you will be prompted to enter the passphrase when loading the binary dump.

Notes

- If a field needs to be compared to a string containing other than alpha characters (including spaces or numbers), single quote the string. For example: 'John Smith', '11-14-2001', or '25 Main St'.
- Dumping the data using a character comparison is case-sensitive.
- If specifying a monetary value, do not include the money unit. For example, instead of $5.50, enter 5.5.
- If specifying a negative number, escape the minus sign with a backslash (\). For example: proutil testdb -C dumpspecified "PUB.Item.Item-NUM" LT "\-90".
- Dates must be entered as *mm-dd-yyyy*.
- The value entered for *field-name* must be the name of an index field.

- You cannot use DUMPSPECIFIED to dump protected audit data. For more information on auditing and utility security and restrictions, see Auditing impact on database utilities on page 193.
- When dumping the contents of a table encrypted with Transparent Data Encryption, the output of the binary dump is **not** encrypted by default. Encrypt the dump file (container security) by specifying either `-Cipher 11` or `-Cipher 12` and entering a passphrase for the PBE cipher. You will be prompted to enter the passphrase when loading the contents of the dump file.
- When dumping the contents of a multi-tenant database, if no tenant or group is specified, then all the rows meeting the specification are dumped.
- When dumping the contents of a table partitioned table, if no partition is specified, then the entire table is dumped.
- When dumping the contents of a multi-tenant or table-partitioned database, and a tenant, group, partition, or composite initial is specified, the naming convention of the dump file is modified as shown:

```
table-name_ID{T|G|P|C}_TPGCName.bdn
```

The elements of the file name are described in Table 101: Binary dump file name description on page 691.

PROUTIL ENABLEAUDITING qualifier

Enables auditing for a database.

Syntax

```
proutil db-name -C enableauditing area Area-Name
       [indexarea Index-Area-Name ] [deactivateidx ]
```

Parameters

db-name

Name of the database to be enabled for auditing.

`area` *Area-Name*

Name of the area where the auditing data tables are located. If *Area-Name* contains spaces, you must enclose the name in quotation marks.

`indexarea` *Index-Area-Name*

Name of the area where the indexes for the auditing data tables are located. If *Index-Area-Name* contains spaces, you must enclose the name in quotation marks.

`deactivateidx`

Deactivate all non-primary indexes for the auditing tables.

Ensuring the security of your audit data requires that the audit data tables be **empty** when you enable auditing. When you enable your database for auditing, your database can be in one of two possible states: auditing is disabled (or has never been previously enabled) or auditing is deactivated.

If your database has auditing **disabled**, successfully enabling the database results in the following message:

```
Auditing has been enabled for database db-name. (12479)
```

If your database is **disabled**, and your audit data tables are not empty, the following error message is displayed:

```
Auditing can not be enabled because auditing data tables are not empty. (13650)
```

If your database has auditing **deactivated**, successfully enabling the database results in the following message:

```
Auditing has been activated
```

For more information on auditing states, see Auditing states on page 182.

Notes

- When auditing is first enabled for a database, the meta-schema to support auditing is added to the database, and the event table is populated.
- The database must be offline when enabling auditing.
- While not required, it is recommended that the auditing tables and indexes be placed in their own area.
- Deactivating non-primary indexes is available as an option to improve performance when using database auditing. Deactivated indexes may be activated using PROUTIL IDXBUILD. The non-primary indexes are useful for reporting and should be activated on your audit archive database.
- See Auditing tables on page 185 for brief descriptions of the meta-schema tables created when auditing is enabled and the indexes that can be deactivated. For detailed information, see *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

PROUTIL ENABLEAUTHGATEWAY qualifier

Enables the OpenEdge Authentication Gateway for a database.

Syntax

```
proutil db-name -C enableauthgateway
```

Parameters

db-name

Name of the database to be enabled for use of the OpenEdge Authentication Gateway.

PROUTIL ENABLEAUTHGATEWAY enables an OpenEdge database to use the OpenEdge Authentication Gateway for authentication.

Enabling the database validates that there is at least one valid enabled domain to avoid lockout of all logins. If there is no valid domain, ENABLEUATHGATEWAY fails with the following error:

```
Enable failed due to lack of a valid enabled domain.
```

Additionally, there must a valid URL specified in the `_db-option` table. Set the URL with the STS URL Utility. For details, see the STS URL Utility in *OpenEdge Getting Started: Authentication Gateway Guide*. If the base URL name is not set correctly, enablement fails with the following error:

```
Enable failed due to lack of a valid STS URL.
```

For the most secure deployment, Progress Software Corporation recommends that connection role enforcement is also enabled on the database. If the database is NOT enabled for connection role enforcement when enabling the OpenEdge Authentication Gateway, then the following warning is reported:

```
Warning: It is recommended that connection role security enforcement be configured
when using the OpenEdge Authentication Gateway.
```

When successfully enabled, ENABLEAUTHGATEWAY reports:

```
Authentication Validation has been enabled for the database dbname
```

If auditing is enabled, Audit event 11901 is recorded.

PROUTIL ENABLEB2 qualifier

Designates an area for alternate buffer pool processing.

Syntax

```
proutil db-name -C enableB2 area-name
```

Parameters

db-name

Name of the database where you are specifying an area for the alternate buffer pool.

area-name

Name of the area (Type I or Type II) you are designating for the alternate buffer pool.

The Alternate buffer pool is a pool of buffers maintained independently from the primary buffer pool. Before the pool is filled, the buffers are maintained in a first-in-first-out (FIFO) queue. Once the alternate buffer pool is filled, an LRU chain replacement maintains the buffers, however the alternate buffer pool LRU chain is separate from the primary buffer pool LRU chain. Designating "hot" tables or areas to the alternate buffer pool, may increase buffer hit ratios.

Designating an area to participate in the alternate buffer pool requires that an alternate buffer pool is created with the `-B2` startup parameter. If there is no alternate buffer pool, the area consumes buffers from the primary buffer pool.

Notes

- ENABLEB2 must be run offline.
- ENABLEB2 and alternate buffer pool support, are only supported with an Enterprise database license.
- To remove an area from the alternate buffer pool, see PROUTIL DISABLEB2 qualifier on page 683.
- To view the status of areas, see PROUTIL VIEWB2 qualifier on page 778.
- For Type II areas, you can refine your alternate buffer pool designation to the object (table, index, LOB) level through the Data Dictionary and Data Administration. Object-level designations for the alternate buffer pool can be made online.

PROUTIL ENABLECDC qualifier

Enables your database for Change Data Capture (CDC).

Syntax

```
proutil db-name -C enablecdc
     area table-areaname indexarea index-areaname
     [ deactivateidx ]
```

Parameters

db-name

The name of the database where you are enabling CDC.

area*table-areaname*

The name of the area where your Change Tracking Table (_Cdc-Change-Tracking) will reside. The area must be a Type II area, and must be created prior to enabling CDC. Progress Software recommends that you create an area exclusively for the Change Tracking Table.

indexarea*index-areaname*

The name of the area where the indexes of your Change Tracking Table will reside. The area must be a Type II area, and must be created prior to enabling CDC. Four indexes are created for the Change Tracking Table (_Change-Sequence (primary), _Part-Rec-Id, _Policy-Id, _Tran-Id). Progress Software recommends that you create an area exclusively for the Change Tracking Table indexes.

deactivateidx

Add deactivateidx to disable the non-primary indexes of the Change Tracking Table. If deactivated during this enablement, the indexes can be activated at a later time with one of the index utilities (IDXBUILD, IDXACTIVATE,IDXFIX).

Successful execution of ENABLECDC makes your database ready to create CDC policies for capturing change data.

See:

- PROUTIL DISABLECDC qualifier on page 683

Note

ENABLECDC is an auditable event, if your database is enabled and configured for auditing.

PROUTIL ENABLEENCRYPTION qualifier

Enables your database for transparent data encryption.

Syntax

```
proutil db-name -C enableencryption
       [-Cipher cipher-number] [-Autostart {admin | user} ]
       [-biencryption enable | disable ] [-aiencryption enable | disable]
       [-Passphrase] [ [-userid userid] [-password password] ]
```

Parameters

db-name

Name of the database where you are enabling encryption.

-Cipher *cipher-number*

Specifies the database master policy cipher. If not specified, cipher 1, "AES_CBC_128" is used by default. See PROUTIL EPOLICY INFO qualifier on page 704 for a list of the supported ciphers and their corresponding ID numbers.

-Autostart {admin | user}

Specifies that you will allow your database to autostart authentication to the key store by the specified key store account. If not specified, the database is set to manual startup. Manual startup requires that the user enter the passphrase to authenticate the key store every time the database is opened (by servers, clients, and utilities).

-biencryption enable | disable

Specifies whether your BI is encrypted or not. If not specified, BI encryption is enabled.

-aiencryption enable | disable

Specifies whether your AI is encrypted or not. If not specified, AI encryption is enabled.

Note: Enabling ai or bi encryption when it is already enabled will return the following error:

```
Error - Cannot enable ai or bi encryption when it is already enabled. (19417)
```

The encryption will remain enabled.

`-Passphrase`

Specifies that the user must be prompted for a passphrase to authenticate the key store, prior to executing this command.

`-userid` *`userid`* `-password` *`password`*

Specifies the userid and password of an authenticated database administrator.

Successful execution of ENABLEENCRYPTION creates the database key store and makes the database ready for the creation of encryption policies. **No data is encrypted by running this command.**

The first time ENABLEENCRYPTION is run, you are prompted for several passphrases:

1. The key store administrator passphrase (required).
2. The key store user passphrase (optional).
3. The PBE passphrase for creating the database master key (required if `-Cipher 6` specifies that the default cipher is DEC_CBC_PBE).

Passphrases must conform to the constraints described in the table below.

Table 102: Passphrase constraints

Rule	value
Minimum number of characters	8
Maximum number of characters	1024
Minimum number of numeric characters	1
Minimum number of alpha characters	2
Minimum number of punctuation characters	1
Character set	[a-zA-Z0-9]!@#$%^&*()+-{}[]|\,./<>?;:<space>
First character	(see Character set)
Mixed case alpha required	True
Case sensitive	True

Notes

- After successfully enabling your database for encryption, you can run the ENABLEENCRYPTION command again, **only** for the purpose of changing the status of AI and BI encryption.
- To change settings other than AI and BI encryption, use the PROUTIL EPOLICY MANAGE command. See PROUTIL EPOLICY MANAGE qualifier on page 705 for command syntax and details.
- If after-imaging is enabled, ENABLEENCRYPTION causes an extent switch.

PROUTIL ENABLEJTA qualifier

Enables the database to participate in JTA transactions.

Syntax

```
proutil db-name -C enablejta
```

Parameters

db-name

Name of the database to be enabled for JTA transactions.

Notes

- Databases enabled for JTA transactions must perform crash recovery in multi-user mode.
- For more information on JTA transactions, see Distributed Transaction Processing on page 255 and *OpenEdge Data Management: SQL Development*.

PROUTIL ENABLEKEYEVENTS qualifier

Enables the database for the storage of key events.

Syntax

```
proutil db-name -C enablekeyevents [area-name]
```

Parameters

db-name

Name of the database being enabled for key event storage.

area-name

Name of the area where the key events table is located. If *area-name* is not specified, the key events table, _KeyEvt, is created in the Schema Area.

Notes

- The first time ENABLEKEYEVENTS is run on a database, the _KeyEvt table is created in the area specified by *area-name* (or the Schema Area if not specified). Subsequent enablements ignore the *area-name* parameter.
- The database must be offline to enable key events.

PROUTIL ENABLELARGEKEYS qualifier

Enables large key entry support for a database migrated to Release 10.1B or higher from an earlier version.

```
proutil db-name -C enablelargekeys
```

Parameters

db-name

Specifies the name of database to enable for large key entry support.

Large key entries increase the amount of user data in an index from approximately 200 bytes to approximately 1970 bytes.

Notes

- Large keys are supported on databases with 4K or 8K block sizes.
- Large key support cannot be disabled.
- Large key support can cause run time problems for older clients. A client from 10.1A or earlier cannot update or delete any row or field that is referenced by a large index key. PROUTIL ENABLELARGEKEYS issues a warning and requires confirmation before executing, as shown in the following example:

```
$ proutil docsample -C enablelargekeys
OpenEdge Release 10.1B1P as of Mon Oct 23 19:54:20 EDT 2006

If you enable large key entries support for this database, it may cause runtime issues
 to pre-10.1B clients that connect to this database.
Do you really want to enable large keys support? (y/n) (13929)
y
Large Key Entry Feature enabled for database docsample. (13700)
$
```

- If the database is online when enabling large keys, only newly connected clients will be able to perform operations with large keys.
- The database may be online or offline when enabling large keys.
- Databases with 4K and 8K block sizes, created with Release 10.1B or higher, have large key support by default.
- There is no need to rebuild indexes after enabling large keys.

PROUTIL ENABLEMULTITENANCY qualifier

Enables the support of multi-tenant tables in a database.

Syntax

```
proutil db-name -C enablemultitenancy
```

Parameters

db-name

Name of the database to be enabled for multi-tenant tables.

Multi-tenant tables allow database objects to be separated by tenants and groups.

Notes

- The database may be online or offline when enabling multi-tenancy.
- To enable multi-tenancy, the database must have one of the following license scenarios:
  - OE Enterprise RDBMS and OE Multi-tenant Tables
  - OE Development Server

PROUTIL ENABLEREADONLYPARTITIONS qualifier

Enables a table-partitioned database with the ability to change a partition to a read-only state.

Syntax

```
proutil db-name -C enablereadonlypartitions
```

Parameters

db-name

Name of the database to be enabled for read-only partitions.

The database must be enabled for table partitioning prior to executing ENABLEREADONLYPARTITIONS.

See PROUTIL PARTITIONMANAGE ALTER qualifier on page 746 for information on changing partition states between read-only and read-write.

PROUTIL ENABLESTOREDPROC qualifier

Enables stored procedures and triggers for 64-bit databases created prior to OpenEdge Release 10.1B.

Syntax

```
proutil db-name -C enablestoredproc
```

Parameters

db-name

Name of the database to be enabled for 64-bit stored procedures and triggers.

Notes

- The database may be online or offline when enabling stored procedures and triggers.
- Databases created with Release 10.1B and higher have 64-bit stored procedures and triggers by default.
- 64-bit stored procedures and triggers cannot be disabled.
- Once the 64-bit stored procedures and triggers tables have been added to the database, they cannot be removed. However, the existence of the stored procedures and triggers tables and views are harmless.

PROUTIL ENABLETABLEPARTITIONING qualifier

Enables your database for table partitioning.

Syntax

```
proutil db-name -C enabletablepartitioning
```

Parameters

db-name

The name of the database where you are enabling table partitioning.

Successful execution of ENABLETABLEPARTITIONING makes your database ready to partition existing tables and create new partitioned tables.

Note

You must have the Enterprise RDBMS database and Table Partitioning licenses to partition tables.

PROUTIL EPOLICY INFO qualifier

Displays the valid ciphers for the database master key and object-level encryption and their command line cipher number.

Syntax

```
proutil db-name -C epolicy info [ [-userid userid] [-password password] ]
```

Parameters

db-name

Name of the database.

`-userid` *userid* `-password` *password*

Specifies the userid and password of an authenticated database administrator.

When ciphers are specified on the command line to other utilities, you must identify the cipher by the number ID listed by this utility. The output of EPOLICY INFO follows:

```
----------DMK Cipher Selection-------------
----ID----Cipher---Mode---Size----Key type--
     1    AES      CBC    128     binary
     2    AES      CBC    192     binary
     3    AES      CBC    256     binary
     4    DES      CBC     56     binary
     5    DES3     CBC    168     binary
     6    DES      CBC     56     PBE

----------Object Cipher Selection-----------
----ID----Cipher---Mode---Size----Key type--
     0    NULL     NULL    N/A     N/A
     1    AES      CBC    128     binary
     2    AES      CBC    192     binary
     3    AES      CBC    256     binary
     4    DES      CBC     56     binary
     5    DES3     CBC    168     binary

-----Binary Dump File Cipher Selection------
----ID----Cipher---Mode---Size----Key type--
     6    DES      CBC    56      PBE
    11    AES      CBC    128     PBE
    12    AES      CBC    256     PBE
```

PROUTIL EPOLICY MANAGE qualifier

Manages the encryption policies for the specified database object, or the autostart or key store status.

Syntax

```
proutil db-name -C epolicy manage
      [object-type {encrypt |cipher |rekey|update} object-name ]
     | [keystore {reconstruct |rebind |userphrase |adminprhase} ]
     | [autostart { admin|user |disable} ]
      [ [-userid userid] [-password password] ]
```

Parameters

db-name

Name of the database.

object-type

Specifies the type of database object being managed. Valid object types are: `area`, `index`, `lob`, and `table`.

`encrypt`

Specifies that the action on the object is to encrypt the blocks.

`cipher`

Specifies that the action on the object is to change the cipher. You must have previously encrypted the object.

`rekey`

Specifies that the action on the object is to change the cipher key. You must have previously encrypted the object.

`update`

Specifies that the action on the object is to update all the blocks of the object. Update scans all the blocks in an object and updates the blocks to the current encryption policy.

object-name

Specifies the name of the object identified by *object-type*. For the object-type `area`, the object type must specify a Type I area. For all other object-type values, the specific object must reside in a Type II area.

`keystore reconstruct`

Specifies regeneration of the database key store. You are prompted for the database master key passphrase.

Only a key store with a database master key generated with the DES_CBC_PBE cipher can be reconstructed.

`keystore rebind`

Specifies to rebind a new database guid with database master policy after executing PROCOPY `-newinstance`. You are always prompted for the key store admin passphrase when running this command.

`keystore userphrase`

Specifies to change the key store user passphrase. You are prompted for the user passphrase. A blank passphrase **is** acceptable. The database must be in single-user mode to run this command.

`keystore adminphrase`

Specifies to change the key store admin passphrase. You are prompted for the admin passphrase. A blank passphrase **is not** acceptable. The database must be in single-user mode to run this command.

`autostart user`

Specifies that the database can be started in unattended mode with the key store user account.

`autostart admin`

Specifies that the database can be started in unattended mode with the key store admin account.

`autostart disable`

Specifies that the database cannot be started in unattended mode.

`-userid` *`userid`* `-password` *`password`*

Specifies the userid and password of an authenticated database administrator.

`-Passphrase`

Specifies that the user must be prompted for the key store administrator passphrase to authenticate the key store before running this command.

Notes

- Operations on areas must be performed while the database is offline.
- For details on Transparent Data Encryption, see Transparent Data Encryption on page 199 and *OpenEdge Getting Started: Core Business Services - Security and Auditing*.

PROUTIL EPOLICY SCAN qualifier

Reports on the encryption policies for the specified database object, or the autostart or key store status.

Syntax

```
proutil db-name -C epolicy scan [area area-name | index index-name |
  lob lob-name | table table-name ] [ [-userid userid] [-password password] ]
```

Parameters

`db-name`

Name of the database.

`area` *`area-name`*

Specifies to scan the policies of the *`area-name`* area. You must specify a Type I area to scan; you cannot scan policies for a Type II area.

`index` *`index-name`*

Specifies to scan the policies of the *`index-name`* index.

`lob` *`lob-name`*

Specifies to scan the policies of the *`lob-name`* lob.

`table` *`table-name`*

Specifies to scan the policies of the *`table-name`* table.

`-userid` *`userid`* `-password` *`password`*

Specifies the userid and password of an authenticated database administrator.

EPOLICY SCAN reads each block of an object and reports the numbers of blocks for each encryption policy in the object.

PROUTIL EPOLICY VIEW qualifier

Views the encryption policies for the specified database object, or the autostart or key store status.

Syntax

```
proutil db-name -C epolicy view
       [ area area-name | index index-name | lob lob-name | table table-name
         | keystore | autostart ] [ [-userid userid] [-password password] ]
```

Parameters

`db-name`

Name of the database.

`area` *`area-name`*

Specifies to view the policies of the *`area-name`* area. You must specify a Type I area to view; you cannot view policies for a Type II area. Leave the parameter blank to view a summary of all Type I areas that have encryption specified.

`index` *`index-name`*

Specifies to view the policies of the *`index-name`* index. Leave the parameter blank to view a summary of all indexes that have encryption specified.

`lob` *`lob-name`*

Specifies to view the policies of the *`lob-name`* lob. Leave the parameter blank to view a summary of all lobs that have encryption specified.

`table` *`table-name`*

Specifies to view the policies of the *`table-name`* table. Leave the parameter blank to view a summary of all tables that have encryption specified.

`keystore`

Specifies to view the key store settings.

`autostart`

Specifies to view the current autostart settings.

`-userid` *`userid`* `-password` *`password`*

Specifies the userid and password of an authenticated database administrator.

When you specify a database object (area, index, table, lob) EPOLICY VIEW reports on the encryption policy status of the object, including the current and previous policies (if it exists).

When you specify the key store, EPOLICY VIEW reports on the state of the key store, including: key store version, key store creation date, key store update date, account status, and information about the database master key.

When you specify autostart, EPOLICY VIEW reports on the current autostart setting.

Note

Encryption policies are not immediately applied to ai areas that are either busy or full at the time ai encryption is enabled. As such, EPOLICY VIEW may return a report featuring a mix of results for newly encrypted ai areas in which some areas are enabled for encryption and some are not. Once all ai areas are emptied, encryption will be applied normally, and EPOLICY VIEW will return a report detailing the applied encryption policy.

PROUTIL HOLDER qualifier

Determines whether the database currently in use is in single-user mode, multi-user mode, or in use by a utility.

```
proutil db-name -C holder
```

Parameters

`db-name`

Specifies the database you are using.

Notes

- When the HOLDER qualifier completes, it sets a return code that you can test in a UNIX script or Windows batch file. This information is useful to obtain before performing a database backup or shutting down a database. The return codes for the HOLDER qualifier on UNIX are shown in the following table:

Return code	Description
0	The database is not in use
14	The Database is locked by single user, PROUTIL, or RFUTIL
16	The database is open in multi-user mode

  Any other nonzero return code indicates that an error has occurred.

- Return codes can be added or changed from one release to the next. Use scripts that depend on a specific return code with caution.

- This example shows how you might use the HOLDER qualifier on UNIX in a script that performs a database backup:

```
proutil mydb -C holder
retcode=$?  # this saves the return code
case $retcode in
0) echo "The database is not busy, ok to backup"
   ;;
14) echo "The database is locked, no backup done"
   exit $retcode
   ;;
16) echo "The database is busy in multi-user mode, no backup done"
   exit $retcode
   ;;
*) proutil failed, no backup done
   echo error code = $retcode
   exit $retcode
   ;;
esac
proutil mydb -C truncate bi
<<test the return value from proutil here>>
<<put the backup command here>>
rfutil mydb -C mark backedup
```

PROUTIL IDXACTIVATE qualifier

Activates an inactive index.

Syntax

```
proutil db-name -C idxactivate [owner-name.] table-name.index-name
          [useindex index-name] [recs n] [refresh t]
          [ tenant tenant-name | group group-name |
             partition partition-name | composite initial ]
```

Parameters

db-name

Specifies the database containing the index being activated.

[*owner-name.* **]** *table-name.index-name*

Specifies the index being activated. You must specify the table and index names. Specify the owner name if tables in your database have more than one owner.

useindex *index-name*

Specifies an index to use while accessing the records in your table to build the index being activated. This index must be active and cannot be a word index. If you omit this parameter or specify a word index, the primary index is used.

Caution:

If the index you specify is not unique, a stricter locking protocol must be used when activating the index, causing an increase in time to execute IDXACTIVATE.

If you select a non-unique index, you are warned and asked to confirm your choice as follows:

```
A non-unique index has been selected as the useindex index.
Additional locking is required with the use of this index
<index-name>.
Do you want to continue (y/n)?
```

`recs` *n*

Specifies the number of records to process in one transaction. If you omit this parameter, IDXACTIVATE will process 100 records per transaction by default.

`refresh` *t*

Specifies the frequency in seconds to update the display of clients that are blocking the index activation. The default refresh rate is 60 seconds. You can set it as high as 300 seconds. Connected clients with a schema timestamp earlier than the index's schema timestamp will prevent activation of the index.

`tenant` *tenant-name*

Specifies that the IDXACTIVATE operation is to be performed **only** on the index partitions of *tenant-name*. If *tenant-name* does not own any partition of the specified indexes, IDXACTIVATE exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *group-name*

Specifies that the IDXACTIVATE operation is to be performed **only** on the index partitions of *group-name*. If *group-name* does not own any partition of the specified indexes, IDXACTIVATE exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`partition` *partition-name*

Specifies that the IDXACTIVATE operation is to be performed **only** on the index partition of *partition-name*. If *index-name* is not a local index, or the partition cannot be found for the specified table, IDXACTIVATE exits with an error. Specifying `partition` is only allowed on databases enabled for table partitioning.

`composite initial`

Specifies that the IDXACTIVATE operation is to be performed **only** on the index partition of the composite initial partition. If *index-name* is not a local index, or the partition cannot be found for the specified table, IDXACTIVATE exits with an error. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

Notes

- IDXACTIVATE works for both online and offline databases.

- If IDXACTIVATE detects clients with an earlier schema timestamp than the index, there are two possible outcomes:
  - For OpenEdge Release 11.7 and later, if you have enabled Database Client Notification (`-usernotifytime`), IDXACTIVATE waits until all connected clients respond to the notification, and then proceeds.
  - For releases prior to Release 11.7, or if you have not enabled Database Client Notification (`-usernotifytime 0`), you may wait for those clients to disconnect, SQL clients to start a new transaction, or forcibly disconnect them with PROSHUT. IDXACTIVATE can not complete and activate the index until the clients with an earlier schema timestamp are disconnected from the database.
- For partitioned tables, IDXACTIVATE only activates a local index partition of one table partition at a time. You can run multiple IDXACTIVATE processes in parallel to activate indexes for multiple partitions of the same local index.
- For global indexes of a partitioned table, you can not specify `partition` *`partition-name`* or `composite initial`. The entire index must be activated.
- If *`db-name`* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and IDXACTIVATE exits.
- You cannot activate indexes on schema or auditing tables with IDXACTIVATE.

PROUTIL IDXANALYS qualifier

Displays information on index blocks.

Syntax

```
proutil db-name -C idxanalys [ area area-name ]
     [ tenant tenant-name | group group-name | shared ]
     [-csoutput [ -fieldsep sep-value] [ -csfilePrefix pref-name ]
          [ -verbose ] ]
```

Parameters

`db-name`

Specifies the database to analyze.

`area` *`area-name`*

Specifies that IDXANALYS is to be performed on the *`area-name`* area only.

`tenant` *`tenant-name`*

Specifies that IDXANALYS is to be filtered to only provide information related to objects that are owned by *`tenant-name`*. If *`tenant-name`* is not valid, IDXANALYS exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *`group-name`*

Specifies that IDXANALYS is to be filtered to only provide information related to objects that are owned by *`group-name`*. If *`group-name`* is not valid, IDXANALYS exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`shared`

Specifies that IDXANALYS is to be filtered to only provide information related to shared objects. Specifying `shared` is only allowed on databases enabled for multi-tenancy.

`-csoutput`

Specifies that IDXANALYS output the analysis data to text files in addition to the screen. The result is that the following files are created:

- *`db-name`*`.ix.txt` — For index analysis output
- *`db-name`*`.block.txt` — For block analysis output

For more information about the contents of the files, see the Database analysis output on page 667.

`-fieldsep` *`sep-value`*

When `-csoutput` is specified, *`sep-value`* indicates the value to separate the columns of output. A space is used by default. Accepted values for *`sep-value`* are:

- Any ASCII character between 0x21 and 0x7E
- `sp` — to denote a space (the default)
- `tab` — to denote a the TAB character (`<\t>` or ASCII character 0x09)

`-csfilePrefix` *`pref-name`*

When `-csoutput` is specified, *`pref-name`* indicates a file name prefix. The database name is used by default. All remaining elements of the file name are not customizable.

`-verbose`

When `-csoutput` is specified, the first row of the output files contains header names for each column of output.

IDXANALYS calculates the number of index blocks and the percentage of the database used by the index blocks. The `_UserStatus` virtual system table displays the utility's progress. See Managing Performance on page 275 for more information.

Output format

Both non-partitioned and partitioned indexes appear in the summary for shared objects in alphabetical order by name. For partitioned indexes, the data for the index of indexes appears first, followed by the remainder of partitions in alphabetical order by partition name. There is no differentiation between local and global indexes. However, a global index will have only a single line of information with a partitionId of 0.

Example

The following output is a sample display of PROUTIL IDXANALYS:

```
INDEX BLOCK SUMMARY FOR AREA "Employee": 7
------------------------------------------------------


INDEX BLOCK SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
Table                       Index  Fields Levels         Blocks    Size  % Util  Factor
PUB.Benefits
  EmpNo                         8       1      1              1  197.0B     4.8     1.0
PUB.Department
  DeptCode                     18       1      1              1   84.0B     2.1     1.0
PUB.Employee
  DeptCode                     20       1      1              1  132.0B     3.2     1.0

  EmpNo                        19       1      1              1  503.0B    12.4     1.0

  Name                         21       2      1              1    1.0K    25.5     1.0
PUB.Family
  EmpNoRelativeName            22       2      1              1    1.4K    36.4     1.0
PUB.TimeSheet
  EmpNoDayRecorded             59       2      1              1  293.0B     7.2     1.0
PUB.Vacation
  EmpNoStartDate               60       2      1              1  152.0B     3.7     1.0


                    ---------------------------------------------------------------------
Subtotals:                                                    8    3.8K    11.9     0.0



INDEX BLOCK SUMMARY FOR AREA "Inventory": 8
------------------------------------------------------


INDEX BLOCK SUMMARY FOR SHARED OBJECTS:
-------------------------------------------
Table                       Index  Fields Levels         Blocks    Size  % Util  Factor
PUB.Bin
  BinNum:0                     10       1      2              3    6.8K    57.1     1.9

  ItemNum:0                    11       1      1              1    1.8K    46.2     1.0
PUB.Item
  CatDescription               34       1      2              3    4.1K    34.5     2.3

  Category2ItemName            35       2      1              1  958.0B    23.5     1.0

  CategoryItemName             36       2      1              1    1.0K    26.1     1.0

  ItemName                     37       1      1              1  934.0B    23.0     1.0

  ItemNum                      33       1      1              1  503.0B    12.4     1.0
PUB.POLine
  PONumLinenum                 47       2      2             15   51.3K    86.1     1.3
PUB.PurchaseOrder
  PONum                        48       1      2              6   18.8K    78.9     1.4
PUB.Supplier
  SupplierID                   56       1      1              1   97.0B     2.4     1.0
PUB.SupplierItemXref
  ItemNumSupplierID            58       2      1              1  624.0B    15.3     1.0

  SupplieridItemNum            57       2      1              1  542.0B    13.3     1.0
```

```
PUB.Warehouse
  warehousename               62       1      1              1  307.0B     7.5     1.0
  warehousenum                61       1      1              1  133.0B     3.3     1.0

                         ------------------------------------------------------------
Subtotals:                                                   37   87.9K    59.8     0.2
```

PROUTIL IDXBUILD qualifier

Rebuilds an index, packing or consolidating index records to use disk space as efficiently as possible.

Caution: Back up your database prior to executing IDXBUILD. Under certain conditions, if the index rebuild crashes, the only method of recovery is to restore a backup.

Syntax

```
proutil db-name -C idxbuild
   [ all | activeindexes | inactiveindexes |
     table [owner-name.] table-name
   [ partition partition-name | composite initial] |
     area area-name |  schema schema-owner]
   [ tenant tenant-name| group group-name | shared ]
   [ -thread n ] [ -threadnum n ]
   [ -datascanthreads n ] [ -mergethreads n ]
   [ -T dir-name | -SS sort-file-directory-specification ]
   [ -TB blocksize ] [ -TM n ] [ -TMB n ]
   [ -TF n ] [ -freeTF n ] [ -B n ]
   [ -SG n ]
   [ -pfactor n ] [ -rusage]
```

Parameters

db-name

Specifies the database you are using.

`all`

Specifies that you want to rebuild all your indexes. PROUTIL automatically rebuilds all your indexes without asking about disk space requirements. If you also specify `tenant` or `group` on the command line with `all`, only the indexes belonging to the `tenant` or `group` are rebuilt.

`activeindexes`

Specifies that you want all currently active indexes rebuilt.

`inactiveindexes`

Specifies that you want all currently inactive indexes rebuilt. When an inactive index is rebuilt, it becomes active.

`table [` *`owner-name.`* `]` *`table-name`*

Specifies that you want to rebuild the indexes defined for the named table. When you specify `table`, PROUTIL automatically rebuilds the indexes defined for the table without asking about disk space requirements.

`partition` *`partition-name`*

Specifies that the IDXBUILD operation is to be performed only on the indexes of the *`partition-name`* table partition. *`partition-name`* must be a defined partition of `[`*`owner-name.`*`]`*`table-name.`* Specifying `partition` is only allowed on databases enabled for table partitioning.

`composite initial`

Specifies that the IDXBUILD operation is performed on the indexes of the composite initial partition for a partitioned table.

`area` *`area-name`*

Specifies that you want to rebuild all the indexes defined in the named area. When you specify `area`, PROUTIL automatically rebuilds the indexes defined for the area without asking about disk space requirements.

`schema` *`schema-owner`*

Specifies that you want to rebuild all the indexes owned by the named *`schema-owner`*. When you specify `schema`, PROUTIL automatically rebuilds the indexes owned by the *`schema-owner`* without asking about disk space requirements.

`tenant` *`tenant-name`*

Specifies that the IDXBUILD operation is to be performed **only** on the index partitions of *`tenant-name`*. If *`tenant-name`* does not own any partition of the specified indexes, IDXBUILD exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *`group-name`*

Specifies that the IDXBUILD operation is to be performed **only** on the index partitions of *`group-name`*. If *`group-name`* does not own any partition of the specified indexes, IDXBUILD exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`shared`

Specifies that the IDXBUILD operation is to be performed **only** on the shared indexes.

-thread *n*

For Enterprise licenses, IDXBUILD is a multi-threaded operation by default. The -thread parameter allows you to turn off multi-threading. By default, *n* is equal to one (1), indicating that the IDXBUILD is multi-threaded. Setting *n* to zero (0) turns off multi-threading.

-threadnum *n*

For Enterprise licenses, when IDXBUILD is run in a multi-threaded mode, the -threadnum parameter allows you to control the maximum number of threads created. By default, the maximum number of threads, *n*, is equal to the number of CPUs. Setting *n* to zero (0) turns off multi-threading.

-datascanthreads *n*

Specifies the number of concurrent threads used to scan table data associated with the indexes being rebuilt. When specified, **-datascanthreads** invokes a multi-threaded data scan that spawns *n* threads for the data scan phase. The data being scanned must meet certain criteria for a multi-threaded data scan to occur; see the Notes section for details.

-mergethreads *n*

Specifies the number of concurrent threads to spawn during the sort/merge phase of index rebuild that merges the key entries in the sort blocks produced by the data scan phase for an individual index sort group. A value of 0 (the default) results in the original, pre-Release 11.1 behavior of one thread per index sort group. A value of 1 also results in the original behavior of one thread per index sort group but the sort/merge is performed by a newly spawned thread rather than by the thread assigned to the individual index sort group.

-T *dir-name*

Specifies the name of the directory where the temporary files are stored. If you do not use this parameter, PROUTIL places temporary files in the current working directory.

-SS *sort-file-directory-specification*

Identifies the location of a multi-volume sort file specification. If you use the Sort Directory Specification (-SS) parameter, PROUTIL does not use the Temporary Directory (-T) parameter.

-TB *n*

Specifies that the index rebuild will be performed using Speed Sort. *n* indicates the allocated block size, in kilobytes. The maximum value is 64; if not specified, the default is 8.

-TMB *n*

Specified the merge block size. This is the size of the blocks used to merge sorted blocks together. This value can be specified separately from the sort block size (-TB). The default value for -TMB is the -TB value. The minimum value is the -TB value. If -TMB is set to a value less than -TB, index rebuild will exit.

-TM *n*

Specifies the merge number. *n* indicates the number of blocks or streams to be merged during the sort process.

-TF *n*

Specifies the memory usage factor. This is the percentage of system memory that can be allocated for merging sort buffers when sorting index keys during the index rebuild process. In the past, there was only one sort merge buffer, and when the sort merge buffer became full, the contents were written to the file system. The default value of -TF is 50 (for 50%). Setting -TF 0 results in the pre-10.2B04 behavior.

Caution: If the memory configuration limits for the process do not allow access to the memory requested, an unpredictable failure may occur, such as being unable to spawn helper threads or other memory allocation failures. Such an error will cause index rebuild to terminate. It is therefore imperative to set the ulimit values appropriately for the index rebuild process.

-freeTF *n*

Specifies whether to free the merge buffer memory explicitly or leave the free to the operating system at the completion of IDXBUILD.

Value of n	Action
0	Do not explicitly free merge buffer memory. The default for Windows 64-bit.
1	Explicitly free merge buffer memory. The default for all Unix platforms and Windows 32-bit.

To improve performance, the merge buffers in the -TF pool are pointer swapped rather than using strcpy(), which results in the list of allocated memory chunks being in sort order rather than allocated order. When the buffers in the -TF pool are freed in an order other than they are allocated, there can be severe performance problems on Windows 64-bit. By setting -freeTF 0 on Windows 64-bit, the allocated memory is freed by the operating system when IDXBUILD completes.

-B *n*

Specifies the number of blocks in the database buffers.

-SG *n*

S

specifies that the index rebuild uses index grouping. *n* must be a value between 8 and 64, and indicates the number of index groups used by IDXBUILD. The default value is 48. Note that a temporary file is created for each index group.

A large -SG value requires more memory allocation and more file handles. To determine the amount of memory (in kilobytes) needed for each index group, add 1 to the merge number (the value of -TM) and multiply the sum by the speed sort block size (the value of -TB). Memory consumption for each index group equals (-TM + 1) * -TB.

-pfactor *n*

Specifies that IDXBUILD will use a packing factor. *n* must be a value between 60 and 100, and indicates the maximum percentage of space used in an index block. The default value is 100.

-rusage

Specifies that IDXBUILD report resource statistics. When specified, index rebuild reports resource statistics including user and system CPU usage and approximate disk reads and write I/O statistics.

If you do not specify all, table, area, or schema, the following menu appears:

```
Index Rebuild Utility
=====================

Select one of the following:
All            (a/A) - Rebuild all the indexes
Some           (s/S) - Rebuild only some of the indexes
By Area        (r/R) - Rebuild indexes in selected areas
By Schema      (c/C) - Rebuild indexes  by schema owners
By Table       (t/T) - Rebuild indexes in selected tables
By Partition   (p/P) - Choose indexes in selected table partitions
By Activation (v/V) - Rebuild selected active or inactive indexes
-------------------
Multi-Tenancy (m/M) - Choose tenants or groups
-------------------
Quit      (q/Q) - Quit, do not rebuild

Enter your selection:
```

Note: Depending on the features enabled on your database, some menu items do not appear.

The table below describes PROUTIL IDXBUILD options.

Table 103: PROUTIL IDXBUILD options

Option	Action
All	Prompts you to verify whether you have enough disk space for index sorting.
Some	Prompts you for the indexes you want to rebuild by first entering the table name, and then the index name. IDXBUILD then prompts you to verify whether you have enough disk space for index sorting.
By Area	Prompts you for the area containing the indexes you want to rebuild, then prompts you for the indexes in the area, then prompts you to verify whether you have enough disk space for index sorting.
By Schema	Prompts you for the schema owner of the indexes you want to rebuild, then prompts you for the indexes, then prompts you to verify whether you have enough disk space for index sorting.
By Table	Prompts you for the table containing the indexes you want to rebuild, then prompts you for the indexes, then prompts you to verify whether you have enough disk space for index sorting.
By Partition	Prompts you for the table partitions containing the indexes you want to rebuild, then prompts you for the indexes, then prompts you to verify whether you have enough disk space for index sorting.

Option	Action
By Activation	Prompts you to choose active or inactive indexes, then prompts you for the indexes, then prompts you to verify whether you have enough disk space for index sorting.
Multi- Tenancy	Prompts you to choose tenants or groups, and then prompts you for the specific tenants or groups. This menu option is only available for databases enabled for multi-tenancy when a `tenant`, `group`, or `shared` was not specified on the command line.
Quit	Quits without rebuilding any indexes.

In addition to rebuilding an index, IDXBUILD also:

- Compresses index blocks to minimize space usage
- Activates deactivated indexes in the database.
- Repairs corrupted indexes in the database (index corruption is typically signaled by error messages)

Notes

- Use the Temporary Directory (`-T`) startup parameter to identify or redirect temporary files created by the PROUTIL utility to a specified directory when sorting and handling space issues.
- Use the Speed Sort (`-TB`), Merge Number (`-TM`), Sort Grouping (`-SG`) and Blocks in Database Buffers (`-B`) startup parameters to improve index rebuild performance.
- IDXBUILD does not repair corrupted record data.
- Use the following formulas to determine whether you have enough free space to sort the indexes:
  - If you rebuild all the indexes in your database, sorting the indexes requires free space that can be as much as 75 percent of the total database size.
  - If you rebuild an individual index, sorting that index requires free space that can be as large as three times the size of one index entry times the number of records in the file.
- The Index Rebuild utility rebuilds an index or a set of indexes in three phases:
  - The utility scans the database by area, clearing all index blocks that belong to the indexes and rebuilds and adds those blocks to the free-block list.
  - The utility scans the database by area and rebuilds all the index entries for every data record. If you chose to sort the index, the utility writes the index entries to the sort file. Otherwise, the utility writes the index entries to the appropriate index at this point.
  - If you indicated that you have enough space to sort the indexes, the utility compacts the index by sorting the index entries in the sort file into groups and entering those entries in one index at a time.
- See Managing Performance on page 275 for more information about using the IDXBUILD qualifier.
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and IDXBUILD exits.
- During the index rebuild, the `_index` records for the indexes to be rebuilt are first changed to "inactive". After the rebuild is complete, they are changed to "active".
- If there is a audit policy defined to record updates on `_index` records for the indexes to be rebuilt, auditing does not record updates. The reason is, if the indexes are in a rebuild state, the indexes used to get auditing policies might be invalid and the schema and policy records needed to record the event may not be accessible.

- For databases enabled for multi-tenancy, if you specify a `tenant` or `group` only those indexes belonging to the tenant or group are rebuilt. If you specify `shared`, only shared indexes are rebuilt.
- For databases enabled for table partitioning, the **Choose Indexes** screen provides an additional column to identify indexes. A partitioned table can have global (**G**) and local(**L**) indexes, as indicated in the **G/L** column. If the column is blank, the index does not belong to a partitioned table.
- For databases enabled for table partitioning, index partition selection is governed as follows:
  - When **By Table** is first selected, if a table and local index are selected, all partitions are selected. You can not select individual partitions.
  - When **By Area** is first selected, only the index partitions in the selected areas are available for selection. Consequently if a local index has multiple partitions in multiple areas, only the index partitions in the selected areas are selectable.
  - When **By Schema** is selected, all partitions of the local indexes are available for selection.
  - When **By Partition** is selected, only local indexes belonging to the selected partitions are available for selection.
  - When the **Some** option is selected, if a local index is entered, you are warned that the index is partition-aligned, and prompted to enter a partition or continue without specifying a partition. If you do not specify a partition, all partitions of the index are processed.
- Use IDXFIX to repair and IDXCOMPACT to compress online.
- For databases enabled for multi-tenancy, you can use MTIDXBUILD to rebuild an index for a tenant or group online.
- For databases enabled for table partitioning, you can use TPIDXBUILD to rebuild an index for a partition online.
- When `-datascanthreads` is specified, it invokes a multi-threaded data scan that spawns *n* threads for the data scan phase. The data being scanned, must meet all of the following criteria for a multi-threaded data scan to occur:
  - The data area being scanned must be a Type II area
  - No index within the data area being scanned is also being rebuilt
  - No index associated with the data area being scanned is a word index
  - Index rebuild must be run with the "sort" option, that is answering "`y`" when asked if you have enough room for sorting.

    If any requirement is not met, the data scan is performed using the original single threaded mechanism. If not specified, the `-datascanthreads` value defaults to 0, also indicating the original single threaded mechanism.

    You can confirm that the data scan phase is multi-threaded by observing the following messages during the data scan phase of index rebuild, as shown:

```
Processing area 689 : (11463)
Start 4 threads for the area. (14536)
Area 689: Multi-threaded record scan (Type II) complete.
Elapsed time: 6.740
```

- Prior to Release 11.1, the threading during the sort/merge phase of index rebuild uses one thread per index sort group by setting the `-thread 1` and `-threadnum` *n* parameters. Specifying `-mergethreads` allows multiple threads to merge data within an individual index sort group. The number of threads spawned during

this phase is the number of merge threads multiplied by the lesser of the -threadnum value or the number of index sort groups in use per area.

There is no restriction on area or index type for invoking the -mergethreads behavior. The index rebuild process must be run with the sort option however since when running index rebuild without the sort option there is no sort/merge phase, and -thread 1 and -threadnum *n* must also be specified.

You can confirm that the sort/merge phase of index rebuild is processing with the new merge threads option by observing the following messages during the sort/merge phase if an index rebuild:

```
Spawning n threads for merging of group n.
```

Note: If your index is small, you may not see the spawning message.

- For both -datascanthreads and -mergethreads, if the requested number of threads can not be spawned, index rebuild exits with an error, for example:

```
Unable to run with requested thread configuration of 48 threads due to
system memory restrictions.
Additional memory required: 100663296 Limit: 134217728 In use: 41036736
Threads requested: Data scan: 0, sort: 8, merge: 5, sort groups: 47
```

PROUTIL IDXCHECK qualifier

Checks database indexes to determine whether an index is corrupt, and if it is, diagnoses the problem.

Syntax

```
proutil db-name -C idxcheck
     [ all |
     table [owner-name.] table-name
              [partition partition-name | composite initial] |
     area area-name | schema schema-owner]
     [ tenant tenant-name | group group-name | shared ]
```

Parameters

db-name

Specifies the database whose index you are checking.

all

Specifies that you want to check all your indexes. If you also specify tenant or group on the command line with all, only the indexes belonging to the tenant or group are checked.

`table [owner-name.] table-name`

Specifies that you want to check the indexes defined for the named table.

`partition partition-name`

Specifies that the IDXCHECK operation is performed only on the indexes of the *partition-name* table partition. *partition-name* must be a defined partition of [*owner-name.*]*table-name*. Specifying `partition` is only allowed on databases enabled for table partitioning.

`composite initial`

Specifies that the IDXCHECK operation is performed on the indexes of the initial partition for a partitioned table.

`area area-name`

Specifies that you want to check all the indexes defined in the named area.

`schema schema-owner`

Specifies that you want to check all the indexes owned by the named *schema-owner*.

`tenant tenant-name`

Specifies that the IDXCHECK operation is to be performed **only** on the index partitions of *tenant-name*. If *tenant-name* does not own any partition of the specified indexes, IDXCHECK exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group group-name`

Specifies that the IDXCHECK operation is to be performed **only** on the index partitions of *group-name*. If *group-name* does not own any partition of the specified indexes, IDXCHECK exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`shared`

Specifies that the IDXCHECK operation is to be performed **only** on the shared indexes.

If you do not specify `all`, `table`, `area`, or `schema`, the following menu appears:

```
Index Check Utility
===================

Select one of the following:
All           (a/A) - Check all the indexes
Some          (s/S) - Check only some of the indexes
By Area       (r/R) - Check indexes in selected areas
By Schema     (c/C) - Check indexes by schema owners
By Table      (t/T) - Check indexes in selected tables
By Partition  (p/P) - Choose indexes in selected table partitions
-------------------
Validation    (o/O) - Change validation options
Multi-Tenancy (m/M) - Choose tenants or groups
-------------------
Quit          (q/Q) - Quit, do not Check

Enter your selection:
```

Note: Depending on the features enabled on your database, some menu items do not appear.

The table below describes the PROUTIL IDXCHECK options.

Table 104: PROUTIL IDXCHECK options

Option	Action
All	Checks all the indexes
Some	Prompts you for the indexes you want to check by first entering the table name, and then the index name. If the table name entered is for a partitioned table and the index name is for a partitioned (local) index, IDXCHECK then prompts for a partition name.
By Area	Prompts you for the area containing the indexes you want to check
By Schema	Prompts you for the schema owner of the indexes you want to check
By Table	Prompts you for the table containing the indexes you want to check
By Partition	Prompts you for the table partitions containing the indexes you want to check.
Validation	Change the validation options
Multi-Tenancy	Prompts you to choose tenants or groups, and then prompts you for the specific tenants or groups. This menu option is only available for databases enabled for multi-tenancy when a `tenant`, `group`, or `shared` was not specified on the command line.
Quit	Quits without checking any indexes

PROUTIL IDXCHECK lets you know whether you have to perform an index rebuild before actually running PROUTIL IDXBUILD. IDXCHECK performs the following operations for each index it checks:

- Reads the contents of the index and the contents of the file, verifies that all the records are indexed, and verifies that each value in the index is in the associated record
- Performs various checks on the data structures in the index to verify that there is no corruption

If PROUTIL IDXCHECK completes successfully, it ensures that all `FIND`, `CAN-FIND`, `GET`, `FOR EACH`, and `PRESELECT` statements that use those indexes work correctly. If errors result, the problem index might produce unexpected results with those statements.

You can modify the validations performed by PROUTIL IDXCHECK. Selecting **Validation (o/O)** from the main menu, presents you with the following menu:

```
Validation Options
----------------
* 1 - Validate physical consistency of index blocks
* 2 - Validate keys for each record
  3 - Validate record for each key
* 4 - Validate key order
  5 - Validate tree
  6 - Validate index against free cluster
  L - Lock tables during the check
  R - Reset error limit, current: 500
  C - Continue to execute
  Q - Quit
```

The options with asterisks are selected by default.

The table below describes the PROUTIL IDXCHECK VALIDATION options.

Table 105: PROUTIL IDXCHECK VALIDATION options

Option	Action
1	Validates physical consistency of index blocks
2	Validates keys for each record
3	Validates record for each key
4	Validates key order
5	Validates the tree. Detects a vulnerability with indexes that span the 32/64 bit boundary within your index area.
6	Validates indexes against the free cluster. Detects if index entries for the special schemas have been incorrectly deleted by PROUTIL MVSCH. These errors only exist if you executed PROUTIL MVSCH on your database with OpenEdge Releases between 11.6.0 and 11.6.3 (with no hotfix). **Warning:** If any errors are reported, contact Progress Technical Support for guidance on a recovery strategy for your database.
L	Locks the tables during the check **Note:** This option is only available when the database is online
R	Resets the error limit; this value is set to 500
C	Continues to execute the check
Quit	Quits without checking any indexes

Notes

- PROUTIL IDXCHECK automatically determines if a server is connected to the database, then decides which mode (online or offline) to run.
- When PROUTIL IDXCHECK is executed on an offline database, no other process can access the database until IDXCHECK completes.
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and IDXCHECK exits.
- When PROUTIL IDXCHECK finds any corruption, it displays error messages. If error messages appear, save the database and the output messages for analysis by Technical Support, back up your database, and then run PROUTIL IDXBUILD.

- IDXCHECK displays error and warning messages on the screen and logs them in the log file. It also displays and logs a success or failure indication, along with the number of errors and warnings issued.
- IDXCHECK might also display warning messages. Although these messages indicate that some condition is a problem, the index is still valid. Check the log file for details.
- See Maintaining Database Structure on page 325 for a description of how to monitor the progress of this utility using the `_UserStatus` virtual system table (VST).
- PROUTIL IDXCHECK uses a temporary file to store the list of indexes being checked during execution of some options. The temporary file is named `<dbname>.<pid>.xb`. The file is deleted when the process successfully completes. If the process terminates unsuccessfully, the file may not be deleted.
- For databases enabled for multi-tenancy, if you specify a `tenant` or `group` only those indexes belonging to the tenant or group are checked. If you specify `shared`, only shared indexes are checked.
- For databases enabled for table partitioning, the **Choose Indexes** screen provides an additional column to identify indexes. A partitioned table can have global (**G**) and local(**L**) indexes, as indicated in the **G/L** column. If the column is blank, the index does not belong to a partitioned table.
- For databases enabled for table partitioning, index partition selection is governed as follows:
  - When **By Table** is first selected, if a table and local index are selected, all partitions are selected. You can not select individual partitions.
  - When **By Area** is first selected, only the index partitions in the selected areas are available for selection. Consequently if a local index has multiple partitions in multiple areas, only the index partitions in the selected areas are selectable.
  - When **By Schema** is selected, all partitions of the local indexes are available for selection.
  - When **By Partition** is selected, only local indexes belonging to the selected partitions are available for selection.
  - When the **Some** option is selected, if a local index is entered, you are warned that the index is partition-aligned, and prompted to enter a partition or continue without specifying a partition. If you do not specify a partition, all partitions of the index are processed.

PROUTIL IDXCOMPACT qualifier

Performs index compaction.

```
proutil db-name -C idxcompact [ owner-name. ] table-name.index-name
   [ tenant tenant-name | group group-name |
        partition partition-name | composite initial ]
 [n]
```

Parameters

db-name

Specifies the source database name.

owner-name

Specifies the owner of the table containing the data you want to dump. You must specify an owner name unless the table's name is unique within the database, or the table is owned by PUB. By default, ABL tables are owned by PUB.

table-name

Specifies the source table containing the source index to be compacted.

index-name

Specifies the source index to be compacted.

tenant *tenant-name*

Specifies that the IDXCOMPACT operation is to be performed **only** on the index partitions of *tenant-name*. If *tenant-name* does not own any partition of the specified index, IDXCOMPACT exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the IDXCOMPACT operation is to be performed **only** on the index partitions of *group-name*. If *group-name* does not own any partition of the specified index, IDXCOMPACT exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

partition *partition-name*

Specifies that the IDXCOMPACT operation is to be performed only on the indexes of the *partition-name* table partition. IDXCOMPACT exits with an error if *partition-name* is not a partition for *table-name*, if *partition-name* is not allocated or in a rebuilding state, or if *index-name* is not a local index for *table-name*. Specifying `partition` is only allowed on databases enabled for table partitioning.

composite initial

Specifies that the IDXCOMPACT operation is performed **only** on the indexes of the initial partition for a partitioned table. IDXCOMPACT exits with an error if the composite initial partition for *table-name* does not exist, if the composite initial partition is not allocated or in a rebuilding state, or if *index-name* is not a local index for *table-name*. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

n

Specifies the degree of index compaction. You can specify an integer >=50 and <=100. The default value is 80. If you do not specify `n`, 80 is used.

Notes

- Index compaction is recommended when the PROUTIL IDXANALYS utility indicates that space utilization of an index is reduced to 60 percent or less. Index compaction increases space utilization of the index block to the compacting percentage specified by *n*.
- Performing index compaction reduces the number of blocks in the B-tree and possibly the number of B-tree levels, which improves query performance.
- The index compacting utility operates in phases:

  - **Phase 1** — If the index is a unique index, the delete chain is scanned and the index blocks are cleaned up by removing deleted entries.
  - **Phase 2** — The nonleaf levels of the B-tree are compacted starting at the root working toward the leaf level.
  - **Phase 3** — The leaf level is compacted.

- PROUTIL IDXCOMPACT can be run either online or offline.
- In addition to compacting an index, this utility clears dead entries left after entries have been deleted from unique indexes.
- If the index specified is a partitioned index, and the `partition` qualifier is not supplied, all partitions of the index are processed. Index partitions that are read-only, rebuilding, or not allocated are skipped.
- Multiple instances of PROUTIL IDXCOMPACT can run simultaneously for different partitions of the same index.
- Because index compacting is performed online, other users can use the index simultaneously for read or write operation with no restrictions. Index compacting only locks one to three index blocks at a time, for a short time, allowing full concurrency.
- The IDXCOMPACT utility does not lock any record or table.
- No other administrative operation on the index is allowed during the compacting process.
- In rare cases where the required percentage of compaction is very high, the compacting percentage might not be reached. Repeating the compacting process a second time might obtain better results.
- See Maintaining Database Structure on page 325 for a description of how to monitor the progress of this utility using the `_UserStatus` virtual system table (VST).

PROUTIL IDXDEACTIVATE

Deactivates an active index or indexes for databases enabled for multi-tenancy or table partitioning.

Syntax

```
proutil db-name -C idxdeactivate
     { table [owner-name.] table-name |
         index [owner-name.] table-name.index-name }
              [ tenant tenant-name | group group-name |
                    partition partition-name | composite initial ]
```

Parameters

db-name

Specifies the database containing the index or indexes being deactivated.

`table` [*owner-name.*] *table-name*

Specifies the table of the index or indexes being deactivated. Specify the owner name if tables in your database have more than one owner.

`index [ owner-name. ] table-name.index-name`

Specifies the index being deactivated. Specify the owner name if tables in your database have more than one owner.

`tenant tenant-name`

Specifies that the IDXDEACTIVATE operation is to be performed **only** on the index partitions of *tenant-name*. If *tenant-name* does not own any partition of the specified indexes, IDXDEACTIVATE exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group group-name`

Specifies that the IDXDEACTIVATE operation is to be performed **only** on the index partitions of *group-name*. If *group-name* does not own any partition of the specified indexes, IDXDEACTIVATE exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`partition partition-name`

Specifies that the IDXDEACTIVATE operation is to be performed **only** on the partition *partition-name*. If the partition named *partition-name* does not exist for the specified table, or the index specified is not a local index, IDXDEACTIVATE exits with an error. Specifying `partition` is only allowed on databases enabled for table partitioning.

`composite initial`

Specifies that the IDXDEACTIVATE operation is to be performed **only** on the composite initial partition. If the composite initial partition does not exist for the specified table, or the index specified is not a local index, IDXDEACTIVATE exits with an error. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

IDXDEACTIVATE deactivates single or multiple indexes.

For multi-tenant tables:

- If you specify a table with the `table` parameter, you must specify a tenant or group, and all indexes of the tenant or group, except the primary index, are deactivated.
- If you specify an index with the `index` parameter, you must specify a tenant or group, and only that specific index partition is deactivated.

For table-partitioned tables:

- If you specify a table with the `table` parameter, you must specify a named partition or the composite initial partition. All local indexes of the partition, except the primary index, are deactivated.
- If you specify an index with the `index` parameter, and you specify a named partition or the composite initial partition, the index must be a local index, and only that specific index partition is deactivated.
- If you specify and index with the `index` parameter, and you do **not** specify a named partition or the composite initial partition, the index must be a global index, and that specific index is deactivated.

Notes

- IDXDEACTIVATE works for both online and offline databases.

PROUTIL IDXFIX qualifier

Checks database records and indexes to determine whether an index is corrupt or a record has a missing or incorrect index. IDXFIX will also repair corrupted indexes.

Syntax

```
proutil db-name -silent -C idxfix
     [ tenant tenant-name | group group-name | shared ]
     [ recs numrecs ]
       -NL -rusage
     [ -userid username [ -password passwd ] ]
```

Parameters

db-name

Specifies the database whose index you are checking.

-silent

Indicates that repetitive messages are not sent to the screen or the log file.

tenant *tenant-nam*

Specifies that the IDXFIX operation is to be performed **only** on the index partitions of *tenant-name*. If *tenant-name* does not own any partition of the specified indexes, IDXFIX exits with an error. Specifying tenant is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the IDXFIX operation is to be performed **only** on the index partitions of *group-name*. If *group-name* does not own any partition of the specified indexes, IDXFIX exits with an error. Specifying group is only allowed on databases enabled for multi-tenancy.

shared

Specifies that the IDXFIX operation is to be performed **only** on the shared indexes.

recs *numrecs*

The number of records to process in one transaction. If not specified, *numrecs* defaults to 100.

-NL

Avoids issuing record locks and index admin locks.

-rusage

When specified, IDXFIX reports resource statistics including user and system CPU usage and approximate disk reads and write I/O statistics, and elapsed time.

-userid *username*

Identifies a user privileged to access protected audit data when executing this utility.

-password *passwd*

Password for the privileged user.

With PROUTIL IDXFIX, you can specify whether to scan the database, the indexes, or both. The Index Fix Utility displays the following menu:

```
                    Index Fix Utility

  1. Scan records for missing index entries with index block validation.
  2. Scan indexes for invalid index entries.
  3. Both 1 and 2 above.
  4. Cross-reference check of multiple indexes for a table.
  5. Build indexes from existing indexes.
  6. Delete one record and its index entries.
  8. Scan records for missing index entries.
  q. Quit

  Enter your selection:
```

The table below describes the PROUTIL IDXFIX options.

Table 106: PROUTIL IDXFIX options

Option	Action
1	Scans the database records for missing or incorrect indexes with index block validation.
2	Scans the index for corrupt index entries. You can specify whether to scan all indexes or a specific set of indexes.
3	Checks the index entries, then checks the database entries.
4	Prompts you for the table and indexes for which you want to run a cross-reference check.For the specified indexes the following processing takes place: • Scan the indexes and each entry in the index.\ • For each index entry, fetch the associated record by ROWID and validate that the fields in the record match the fields used to build the index key • After verifying the individual indexes, an index-to-index comparison verifies that the list of ROWIDs generated by one index exist in the other index(es)

Option	Action
5	Allows you to rebuild multiple indexes based on one known index. At the prompts, enter the following: • At the **Table name:** prompt, enter the name of the table for the indexes. • At the first **Index name:** prompt, enter the name of the **source** index. This is the **good** index to be used for locating the rows of the table. • At subsequent **Index name:** prompts, enter the name of index(es) to rebuild. • Enter **!** when you finish entering index names. The utility asks you to verify your entries, and then rebuilds the specified indexes.
6	Prompts you to specify the RECID of the record you want to delete. Deletes one record and all its indexes from the database. Use this option when a record has damaged indexes. **Note:** You can specify either the area name or area number as the area. If an area is invalid, IDXFIX displays a message declaring the area invalid, then halts the action.
8	Scans records for missing index entries. This option performs the same scan as option **1**, but does not perform the index block validation.
q (formerly 7)	Ends the PROUTIL Index Fix utility.

If you specify option 1, 2, 3 or 8 in the table above, the following menu appears:

```
Select one of the following:
All           (a/A) - Fix all the indexes
Some          (s/S) - Fix only some of the indexes
By Area       (r/R) - Fix indexes in selected areas
By Schema     (c/C) - Fix indexes  by schema owners
By Table      (t/T) - Fix indexes in selected tables
By Partition  (p/P) - Choose indexes in selected table partitions
By Activation (v/V) - Fix selected active or inactive indexes
--------------------
Multi-Tenancy (m/M) - Choose tenants or groups
--------------------
Quit     (q/Q) - Quit, do not fix

Enter your selection:
```

The table below describes the PROUTIL IDXFIX options.

Table 107: PROUTIL IDXFIX scan options

Option	Action
All	Prompts you to verify whether you fix all indexes

Option	Action
Some	Prompts you for the indexes you want to fix by first entering the table name, and then the index name; if the table name entered is for a partitioned table and the index name is for a partitioned (local) index, IDXFIX then prompts for a partition name. IDXFIX then prompts you to verify the action.
By Area	Prompts you for the area containing the indexes you want to fix, and then prompts you for the indexes in the area.
By Schema	Prompts you for the schema owner of the indexes you want to fix, then prompts you for the indexes, then prompts you to verify the action.
By Table	Prompts you for the table containing the indexes you want to fix, then prompts you for the indexes, then prompts you to verify whether you have enough disk space for index sorting
By Partition	Prompts you for the table partitions containing the indexes you want to fix, then prompts you for the indexes, then prompts you to verify whether you have enough disk space for sorting
By Activation	Prompts you to chose active or inactive indexes, then prompts you for the indexes, then prompts you to verify the action.
Multi- Tenancy	Prompts you to choose tenants or groups, and then prompts you for the specific tenants or groups. This menu option is only available for databases enabled for multi-tenancy when `tenant`, `group`, or `shared` was not specified on the command line.
Quit	Quits without fixing any indexes.

Note that only options valid for your database appear.

Notes

- PROUTIL IDXFIX performs the following operations for each index it checks:
  - Reads the contents of the index and the contents of the file, verifies that all the values in the records are indexed, and verifies that each value in the index is in the associated record
  - Performs various checks on the data structures in the index to verify that there is no corruption
- IDXFIX displays error messages on the screen and logs them in the log file. It also displays and logs a success or failure indication, along with the number of errors and warnings issued. IDXFIX might also display warning messages. Although these messages indicate that some condition is a problem, the index is still valid. Check the log file for details.
- Index Fix does not provide a comparison of the index scan and the database scan when you run them online because the database can change during operation.
- Index Fix is designed to wait if a record is in the process of being updated, thus ensuring that it does not incorrectly change a user action in process. However, because changes can occur during the scans, the reported data might not exactly match the database at completion. Index Fix displays this warning when you run it online.
- You can run IDXFIX online or offline.

- Index Fix does not delete or disable indexes, but when you run a full database scan with fix offline and it is successful, it enables a disabled index if no errors are found.
- Enabling indexes online is not advisable because it is not possible to detect changes that are being made to indexes while the process is running.
- While IDXFIX can ensure that an index is complete and correct, it cannot make any improvement to the index's utilization level.
- See Maintaining Database Structure on page 325 for a description of how to monitor the progress of this utility using the `_UserStatus` Virtual System Table (VST).
- IDXFIX requires additional security when auditing is enabled for your database. Only a user with Audit Archiver privilege can run this utility. Depending on how user identification is established for your database, you may need to specify the `-userid` and `-password` parameters to run this utility. For more information on auditing and utility security, see Auditing impact on database utilities on page 193
- During the record scan phase of options 1 and 3, logical verification of the a record's index(es) is performed for all RECIDS in the range. However, if an index for the record resides in another area of the database, physical validation of the index block is not performed. You can use the **Validation** option of PROUTIL IDXCHECK to perform a physical validation of a specific index.
- If an index and table reside in different areas, and if an area number (representing the location of the area table) is provided for `RECID` validation, index blocks residing in another area are not verified.
- PROUTIL IDXFIX uses a temporary file to store the list of indexes being fixed during execution of some options. The temporary file is named *dbname*`.`*pid*`.xb`. The file is deleted when the process successfully completes. If the process terminates unsuccessfully, the file may not be deleted.
- For databases enabled for multi-tenancy, options 4 and 5 of PROUTIL IDXFIX operate on shared tables if `tenant` or `group` is not specified on the command line.
- For databases enabled for table partitioning, the **Choose Indexes** screen provides an additional column to identify indexes. A partitioned table can have global (**G**) and local(**L**) indexes, as indicated in the **G/L** column. If the column is blank, the index does not belong to a partitioned table.
- For databases enabled for table partitioning, index partition selection is governed as follows:
  - When **By Table** is first selected, if a table and local index are selected, all partitions are selected. You can not select individual partitions.
  - When **By Area** is first selected, only the index partitions in the selected areas are available for selection. Consequently if a local index has multiple partitions in multiple areas, only the index partitions in the selected areas are selectable.
  - When **By Schema** is selected, all partitions of the local indexes are available for selection.
  - When **By Partition** is selected, only local indexes belonging to the selected partitions are available for selection.
  - When the **Some** option is selected, if a local index is entered, you are warned that the index is partition-aligned, and prompted to enter a partition or continue without specifying a partition. If you do not specify a partition, all partitions of the index are processed.
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and IDXFIX exits.

PROUTIL IDXMOVE qualifier

Moves an index from one application data area to another while the database remains online.

Syntax

```
proutil db-name -C idxmove
     [ owner-name. ] table-name.index-name area-name
     [ tenant tenant-name | group group-name |
        partition partition-name | composite initial ]
```

Parameters

db-name

Specifies the name of the database containing the table.

owner-name

Specifies the owner of the table containing the data you want to dump. You must specify an owner name unless the table's name is unique within the database, or the table is owned by PUB. By default, ABL tables are owned by PUB.

table-name

Specifies the source table containing the index to be moved.

index-name

Specifies the name of an index to move.

area-name

Specifies the area name of the target application data area into which the index is to be moved. Area names that contain spaces must be quoted. For example, "Area Name."

tenant *tenant-name*

For databases enabled for multi-tenancy, specifies that the partition of the index owned by *tenant-name* is to be moved.

If tenant is specified, and *tenant-name* is not valid, or the index is not a multi-tenant index, IDXMOVE exits with an error.

group *group-name*

For databases enabled for multi-tenancy, specifies that the partition of the index owned by *group-name* is to be moved.

If group is specified, and *group-name* is not valid, or the index is not a multi-tenant index, IDXMOVE exits with an error.

partition *partition-name*

For databases enabled for table partitioning, specifies the named partition of the index to be moved.

If partition is specified, and *partition-name* is not valid, or *index-name* is not a local index, IDXMOVE exits with an error. Specifying partition is only allowed on databases enabled for table partitioning.

`composite initial`

For databases enabled for table partitioning, specifies the composite initial partition of the index is to be moved.

If the composite initial partition for *`table-name`* does not exist, or *`index-name`* is not a local index, IDXMOVE exits with an error. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

The PROUTIL IDXMOVE utility operates in two phases:

- **Phase 1** — The new index is constructed in the new area. The old index remains in the old area, and all users can continue to use the index for read operations.
- **Phase 2** — The old index is killed, and all the blocks of the old index are removed to the free block chain. For a large index, this phase might take a significant amount of time. During this phase all operations on the index are blocked until the new index is available; users accessing the index might experience a freeze in their applications.

Notes

- While you can move indexes online, no writes to the table or its indexes are allowed during the move. The IDXMOVE utility acquires a SHARE lock on the table, which blocks all attempts to modify records in the table. Run the utility during a period when the system is relatively idle or when users are doing work that does not access the table.
- No other administrative operation on the moved index is allowed during the move of the index; it will be blocked. For example, you cannot run an index move utility and at the same time run the index fix or the index compacting utilities on the same index.
- Because the index move utility needs to acquire a share lock on the table, there is a possibility that it will have to wait before it can acquire the lock and start operating.
- You might be able to improve performance by moving indexes that are heavily used to an application data area on a faster disk.
- For multi-tenant indexes, you can not specify both `group` and `tenant`, but you must specify one. The *`area-name`* parameter must specify a Type II area.
- Shared indexes of a multi-tenant database are moved without specifying `tenant` or `group` on the command line.
- For an index of a partitioned table, you must specify `partition`, and the *`area-name`* parameter must specify a Type II area.
- For an index of a partitioned table, you cannot move the index of a partition that is in a rebuilding state.
- The `_UserStatus` virtual system table displays the utility's progress. See Maintaining Database Structure on page 325 for more information.

PROUTIL INCREASETO qualifier

Increase certain server startup parameters while the database remains online.

Syntax

```
proutil db-name -C increaseto [ -B n ] [ -B2 n ] [ -L n ] [ -bibufs n ] [
-aibufs n ]
          [ -Mxs n ] [ -omsize n ] [ -mtmpsize n ] [ -cdcsize n ]
```

Parameters

db-name

Specifies the name of the database to update.

-B

Use -B to specify the new value for the number of blocks in the primary buffer pool.

-B2

Use -B2 to specify the new value for the number of blocks in the Alternate Buffer Pool.

-L

Use -L to specify the new limit of the record locking table.

-bibufs

Use -bibufs to specify the new number of before-image buffers.

-aibufs

Use -aibufs to specify the new number of after-image buffers.

-Mxs

Use -Mxs to specify the size of the shared-memory overflow in kilobytes.

-omsize *n*

Use -omsize to specify the number entries in the secondary storage object cache. At startup, a primary and secondary storage object cache are allocated, each with the number of entries specified by the -omsize startup parameter; INCREASETO only modifies the size of the secondary cache.

-mtpmsize *n*

Use -mtpmsize to specify the number of entries in the multi-tenancy partition cache. At startup, a primary and secondary multi-tenancy partition cache are allocated, each with the number of entries specified by the -mtpmsize startup parameter; INCREASETO only modifies the size of the secondary cache.

-cdcsize *n*

Use -cdcsize to increase the size of the CDC Secondary cache.

INCREASETO enables the identified startup parameters to be increased while the database is online. This improves database resiliency by minimizing planned downtime exclusively for changes to startup parameters, providing a robust, high availability mechanism that allows you to modify commonly used database resources while the database is online.

Notes

- You can execute any number of parameters in a single statement. For example:

```
proutil db-name -C increaseto -B 10240 -L 20480 -bibufs 360 -aibufs 500
```

- If a value specified to increase to is less than the current setting, an error is reported and the value is not changed.
- Only one instance of INCREASETO can connect to an online database at a time.
- Parameters specified using INCREASETO are restricted by currently supported limits. For example, the maximum increase for the number of blocks in the database buffer is limited to 1,000,000,000 for 64-bit platforms. See Database Startup Parameters on page 473 for more information on range and limits of `-B`, `-L`, `-bibufs`, `-aibufs`, and `-Mxs`.
- You must have security privileges on the directly connected database to use INCREASETO.
- All users directly connected to the database must have privileges to read any new shared-memory segments created by INCREASETO. If a user does not have access, INCREASETO will prompt you to disconnect the under-privileged users or abort the INCREASETO.

PROUTIL IOSTATS qualifier

Provides current I/O statistics for an active database.

Syntax

```
proutil db-name -C iostats
```

Parameters

db-name

Specifies the active database where you are running database I/O statistics.

The statistics provided by PROUTIL IOSTATS include buffered, unbuffered, and logical I/O database operations. The statistics are cumulative from database startup, or from the last time the statistics were cleared with PROUTIL ZEROSTATS. For more information resetting the statistics, see PROUTIL ZEROSTATS qualifier on page 782.

Example

The following output is a sample display of PROUTIL IOSTATS:

```
Database name is sports (2524)
FILE          BUFFERED        UNBUFFERED          LOGICAL (2556)
            Writes  Reads    Writes  Reads   Reads  Writes  Extends (2557)
================================================================================
sports.b1        0      0       317    389     388     252        2
sports.d1        1    173        11      0     171      11        0
sports_7.d1    773   6792         0      0    6790     772        0
sports_7.d2    270   1702        10      0    1701     269        4
sports_8.d1    649    715         0      0     713     648        0
sports_8.d2      1      1         0      0       0       0        0
sports_9.d1      1      2         0      0       0       0        0
sports_9.d2      1      1         0      0       0       0        0
sports_10.d1     1      2         0      0       0       0        0
sports_10.d2     1      1         0      0       0       0        0
```

`FILE`

Displays the name of the file for the statistics displayed. The file types can include: database files (`.db` extensions), Before-image files (`.bi` extensions), After-image files (`.ai` extensions), and application data extents (`.d`*n* extensions).

`BUFFERED`

Displays the number of buffered reads and writes to the database file for the associated row.

`UNBUFFERED`

Displays the number of unbuffered reads and writes to the database file for the associated row.

`LOGICAL`

Displays the number of client requests for database read and write operations for the associated row.

Notes

- The statistics are cumulative and are reset at database open or when PROUTIL ZEROSTATS is executed.
- IOSTATS is available only on multi-user databases.
- IOSTATS provides a useful alternative to PROMON when you are only interested in statistics on your database extents.
- You can use IOSTATS to determine if your files are opened in buffered or unbuffered mode.

PROUTIL LOAD qualifier

Performs a binary load of database contents.

Syntax

```
proutil db-name -C load filename
      [ tenant tenant-name | group group-name |
        partition partition-name | composite initial ]
       [ build indexes [ refresh n ] ]
       [ -dumplist dumpfile ] [-TB blocksize] [-TM n]
      [ [-T dir-name] | [-SS sort-file-directory-specification] ]
      [-Cipher cipher-id]
```

Parameters

db-name

Specifies the database where you want to load the data. You must completely define the path.

filename

Specifies a single binary dump file that you want to load. You must completely define the path.

tenant *tenant-name*

Specifies that the LOAD operation is to be performed **only** on the table partitions of *tenant-name*. If *tenant-name* is not a valid tenant, LOAD exits with an error. Specifying tenant is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the LOAD operation is to be performed **only** on the table partitions of *group-name*. If *group-name* is not a valid group, LOAD exits with an error. Specifying group is only allowed on databases enabled for multi-tenancy.

partition *partition-name*

Specifies that the LOAD operation is to be performed **only** for the table partition *partition-name*. If *partition-name* is not valid, LOAD exits with an error. Specifying partition is only allowed on databases enabled for table partitioning.

composite initial

Specifies that the LOAD operation is to be performed into the composite initial partition of a database enabled for table partitioning.

build indexes

Indicates that PROUTIL LOAD will simultaneously build the indexes and perform the load, if the table or partition (for a multi-tenant or partitioned table) is **empty** prior to the start of the load.

If tenant, group, partition, or composite initial is specified, build indexes only builds the index for the specified partition. For partition and composite initial global indexes are not deactivated during the load.

For a partitioned database when `partition` or `composite initial` is not specified, global indexes are deactivated and rebuilt during the load.

`refresh` *n*

Specifies the frequency in seconds to update the display of clients that are blocking the build of an index. The default refresh rate is 60 seconds.You can set it as high as 300 seconds. Connected clients with a schema timestamp earlier than the index's schema timestamp will prevent LOAD from building the index until they are disconnected, unless Database Client Notification is enabled (see notes).

`-dumplist` *dumpfile*

Specifies a file containing a list of fully qualified binary dump files to load. Use the file created by PROUTIL DUMP, or create one. For more information on dump files, see Loading table contents in binary format with PROUTIL on page 427.

`-TB` *blocksize*

Specifies that the index rebuild will be performed using Speed Sort. *blocksize* indicates the allocated block size, in kilobytes.

`-TM` *n*

Specifies the merge number. *n* indicates the number of blocks or streams to be merged during the sort process.

`-T` *dir-name*

Specifies the name of the directory in which the temporary files are stored. If you do not use this parameter, PROUTIL places temporary files in the current working directory.

`-SS` *sort-file-directory-specification*

Identifies the location of a multi-volume sort file specification. If you use the Sort Directory Specification (`-SS`) parameter, PROUTIL does not use the Temporary Directory (`-T`) parameter.

`-Cipher` *cipher-id*

For an Enterprise database enabled for Transparent Data Encryption, specify either `-Cipher 11` or `-Cipher 12` to indicate that the contents of the dump file being loaded is encrypted. You are prompted to enter the passphrase that was specified when the dump file was created.

Note: OpenEdge versions prior to Release 12.0 permitted the use of `-Cipher 6` for an Enterprise database enabled for Transparent Data Encryption. For dump files created by OpenEdge versions prior to Release 12, add `-Cipher 6` to the command line.

Note: If when dumping your data using Release 12, you received a warning as a result of adding `-Cipher 6` to the PROUTIL DUMP command, you will be required to enter `-Cipher 11` to the PROUTIL LOAD command when loading the encrypted dump file.

Notes

- See Dumping and Loading on page 411 for more information about the LOAD qualifier.

- The PROUTIL DUMP and LOAD utilities use cyclic redundancy check (CRC) values to establish the criteria for loading.

  The OpenEdge database provides a flexible storage architecture and the ability to relocate objects, such as tables and indexes, while the database remains online. As a result, when you perform a binary load operation, the table numbers in a binary dump file might not match the table numbers in the target database. Therefore, when you perform a binary load operation, the criteria for loading tables is based solely on cyclic redundancy check (CRC) values, and not table numbers.

  For example, when you dump a table, the PROUTIL utility calculates a CRC value for the table and stores it in the header of the binary dump file. When you load the table, PROUTIL matches the CRC value stored in the header with the CRC value of the target table. The values must match or the load is rejected.

  You can load binary dump files created with a previous version of the PROUTIL DUMP utility, because the current version of PROUTIL LOAD uses the CRC value established when the file was originally dumped. Consequently, the OpenEdge database maintains backwards compatibility.

  However, you cannot use PROUTIL LOAD from Version 8.3 or earlier to load a binary dump file created using the Version 9.0 or later of the PROUTIL DUMP utility. The earlier versions of PROUTIL DUMP and LOAD did not use CRC values to establish the criteria for loading, but instead used other mechanisms, such as:

  - Looking up table RECIDs in a target database using the table number stored in the header of the binary dump file
  - Matching table numbers in the header of the binary dump file with table numbers in a target database
  - Comparing the number of fields in the binary dump file with the number of fields in the target database

- PROUTIL LOAD supports loading binary large objects (BLOBS).
- When using PROUTIL LOAD with the `Build Indexes` option, PROUTIL marks the existing indexes as inactive. Once PROUTIL successfully creates the indexes, it marks the indexes active. This means that if the binary load is aborted for any reason, PROUTIL leaves the table with no active indexes.
- Tables loaded with the `build indexes` option must be empty.
- When using PROUTIL LOAD with the `Build Indexes` option, if clients with an earlier schema timestamp than the index are connected to the database, there are two possible outcomes:

  - For OpenEdge Release 11.7 and later, if you have enabled Database Client Notification (`-usernotifytime`), PROUTIL LOAD waits until all connected clients respond to the notification, and then proceeds.
  - For releases prior to Release 11.7, or if you have not enabled Database Client Notification (`-usernotifytime 0`), you may wait for those clients to disconnect, or SQL clients to start a new transaction, or forcibly disconnect them with PROSHUT. PROUTIL LOAD can not complete and build and activate the index until the clients with an earlier schema timestamp are disconnected from the database.

- You cannot use LOAD to load protected audit data. For more information on auditing and utility security and restrictions, see
- If at the time of creating the contents of the binary dump file was encrypted with `-Cipher 11` or `-Cipher 12`, you must specify `-Cipher 11` or `-Cipher 12` respectively to PROUTIL LOAD to indicate that the file is encrypted. For dump files created by OpenEdge versions prior to Release 12, add `-Cipher 6` to the command line. You are prompted to enter the passphrase entered when the file was created. You cannot load the contents without the passphrase.
- When loading the contents of an encrypted binary dump file, the loaded records are not encrypted unless there is an active encryption policy on the table before the load begins. See for more information on creating an encryption policy.

- The keywords `tenant`, `group`, `partition` and `composite initial` are mutually exclusive.
- You cannot specify the composite initial partition with the `partition` qualifier.
- If a table partition is specified, and the LOAD operation encounters records that do not belong to the partition, they are skipped. The number of skipped records is reported at the end of the load. Index keys are added to the global indexes if `build indexes` is not specified.
- If the LOAD operation is loading a partitioned table, records are loaded into the proper partitions according to the partition definition. Index keys are added to the global indexes if `build indexes` is not specified. If `build indexes` is specified, both local partition indexes and the global indexes are rebuilt.
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and LOAD exits.

PROUTIL MTIDXBUILD qualifier

For databases enabled for multi-tenancy, rebuild an index for a tenant or group, packing or consolidating index records to use disk space as efficiently as possible.

Syntax

```
proutil db-name -C mtidxbuild table [owner-name.] table-name
  [ area area-name | index index-name]
  { tenant tenant-name| group group-name }
  [ -T dir-name | -SS sort-file-directory-specification ]
  [ -TB blocksize ] [ -TM n ] [ -B n ] [ -SG n ]
  [ -pfactor n ]
  [ refresh n ]
```

Parameters

db-name

Specifies the multi-tenant database you are using.

`table` [*owner-name.*] *table-name*

Specifies the table for the index or indexes to be rebuilt.

`area` *area-name*

Specifies that you want to rebuild the indexes for *table-name* defined in the named area.

`index` *index-name*

Specifies one named index for *table-name* to be rebuilt.

`tenant` *`tenant-name`*

Specifies that the MTIDXBUILD operation is to be performed **only** on the index partitions of *`tenant-name`*. If *`tenant-name`* does not own any partition of the specified indexes, MTIDXBUILD exits with an error.

`group` *`group-name`*

Specifies that the MTIDXBUILD operation is to be performed **only** on the index partitions of *`group-name`*. If *`group-name`* does not own any partition of the specified indexes, MTIDXBUILD exits with an error.

`-T` *`dir-name`*

Specifies the name of the directory where the temporary files are stored. If you do not use this parameter, MTIDXBUILD places temporary files in the current working directory.

`-SS` *`sort-file-directory-specification`*

Identifies the location of a multi-volume sort file specification. If you use the Sort Directory Specification (`-SS`) parameter, MTIDXBUILD does not use the Temporary Directory (`-T`) parameter.

`-TB` *`n`*

Specifies that the multi-tenant index rebuild will be performed using Speed Sort. *`n`* indicates the allocated block size, in kilobytes. The maximum value is 64; if not specified, the default is 8.

`-TM` *`n`*

Specifies the merge number. *`n`* indicates the number of blocks or streams to be merged during the sort process.

`-B` *`n`*

Specifies the number of blocks in the database buffers.

`-SG` *`n`*

Specifies that the multi-tenant index rebuild uses index grouping. *`n`* must be a value between 8 and 64, and indicates the number of index groups used by MTIDXBUILD. The default value is 48. Note that a temporary file is created for each index group.

A large `-SG` value requires more memory allocation and more file handles. To determine the amount of memory (in kilobytes) needed for each index group, add 1 to the merge number (the value of `-TM`) and multiply the sum by the speed sort block size (the value of `-TB`). Memory consumption for each index group equals (`-TM` + 1) * `-TB`.

`-pfactor` *`n`*

Specifies that MTIDXBUILD will use a packing factor. *`n`* must be a value between 60 and 100, and indicates the maximum percentage of space used in an index block. The default value is 100.

`refresh` *`n`*

Specifies the frequency in seconds to update the display of clients that are blocking the build of an index. The default refresh rate is 60 seconds.You can set it as high as 300 seconds. Connected clients with a schema timestamp earlier than the index's schema timestamp will prevent MTIDXBUILD from building the index until they are disconnected.

Notes

- If you do not specify an area or a specific index, all indexes associated with the table, for the specified tenant or group, are rebuilt.
- You can run multiple instances of MTIDXBUILD at the same time.
- To avoid name collisions with the `.srt` file when running multiple instances of MTIDXBUILD, the naming convention used for the `.srt` file is *`databasename.tenantname`*`.srt` or *`databasename.groupname`*`.srt`. If a file with that name is not located, MTIDXBUILD then looks for a file named *`databasename`*`.srt`.
- Index partitions that are not allocated can not be rebuilt.
- Use the Temporary Directory (`-T`) startup parameter to identify or redirect temporary files created by MTIDXBUILD to a specified directory when sorting and handling space issues.
- Use the Speed Sort (`-TB`), Merge Number (`-TM`), Sort Grouping (`-SG`) and Blocks in Database Buffers (`-B`) startup parameters to improve index rebuild performance.
- MTIDXBUILD prompts you to confirm you have enough disk space for sorting. If you answer "no", MTIDXBUILD executes without sorting.
- MTIDXBUILD does not repair corrupted record data.
- For differences between MTIDXBUILD and IDXBUILD, see Multi-tenant index rebuild on page 369.
- To determine whether you have enough free space to sort the indexes, consider that when rebuilding an individual index, sorting that index requires free space that can be as large as three times the size of one index entry times the number of records in the file.
- During the multi-tenant index rebuild, the `_index` records for the indexes to be rebuilt are first changed to "active", and the `_storageObject` record for the tenant or group partition, is set to "inactive". This enables an index to be in different states for different partitions.
- If auditing is enabled on your database:
  - If there is an audit policy defined to record updates on `_index` records for the indexes to be rebuilt, auditing does not record updates.
  - MTIDXBUILD uses the same audit event as IDXBUILD.
- To prevent users from accessing a rebuilt index with a stale cursor, ABL clients, whose login tenant or group matches the table MTIDXBUILD is working on, or whose tenant id is a super tenant, with a timestamp older than a current timestamp from MTIDXBUILD, must update their cache or log out.

  If MTIDXBUILD detects one or more these clients, there are two possible outcomes:

  - For OpenEdge Release 11.7 and later, if you have enabled Database Client Notification (`-usernotifytime`), MTIDXBUILD waits until the clients respond to the notification, and then proceeds.
  - For releases prior to Release 11.7, or if you have not enabled Database Client Notification (`-usernotifytime 0`), you may wait for those clients to disconnect, for SQL clients to start a new transaction, or forcibly disconnect them with PROSHUT.

PROUTIL MVSCH qualifier

Frees disk space by moving schema. Primarily used after converting a database to a new release.

Syntax

```
proutil db-name -C mvsch
```

Parameters

db-name

Specifies the name of the converted database.

Notes

- After conversion, it is possible to move data into new areas by using PROUTIL dump and load or bulkload qualifiers, the Database Administration tool, the Database Dictionary, or ABL code. However, areas will continue to hold disk space after the removal of data, because the schema remains. Once the area's schema is moved by using PROUTIL MVSCH, the area can be truncated.
- You must truncate the database's BI file before using PROUTIL MVSCH.
- Always back up the database before using PROUTIL MVSCH.

Caution: PROUTIL with the MVSCH qualifier is a nonrecoverable utility. If the execution fails, you cannot connect to the database.

PROUTIL PARTITIONMANAGE ALTER qualifier

Partition management option to alter the state a partition of a partitioned table.

Syntax

```
proutil db-name -C partitionmanage alter table table-name
  { partition table-partition-name | composite initial }
        state { read-only | read-write }
```

Parameters

db-name

Specifies the database where you are performing partition management.

table *table-name*

Specifies the table for the PARTITIONMANAGE ALTER operation.

partition *table-partition-name*

Specifies the partition to alter.

`composite initial`

Specifies that the composite initial partition of the table is being altered.

`state { read-only | read-write }`

Specifies the new state for ALTER to change the partition to.

The partition can be in one of two states:

- `read-only` — The **data** of the partition can not be changed. Language clients can only read the data. For a list of utility access permitted, see below.
- `read-write` — Language clients can read, write, create, and delete data in the partition (provided that access is not restricted at a higher level, such as user-level security with a `can-*` permission in ABL).

PARTITIONMANAGE ALTER is used to change the state of a partition between `read-write` and `read-only`.

The following operations are permitted on a read-only partition:

- READ the data and indexes
- Perform table-level schema changes:
  - Add fields
  - Add indexes
  - Add triggers
  - Delete indexes
- Execute database utilities that perform physical location changes, for example, PARTITIONMANAGE SPLIT, PARTITIONMANAGE MERGE, and TABLEMOVE.

  PARTITIONMANAGE SPLIT has the following restrictions:
  - Read-Only source to Read-Only target(s) is allowed
  - Read-Write source to Read-Only target(s) is allowed
  - Read-Only source to Read-Write target(s) is **not** allowed

  PARTITIONMANAGE MERGE has the following restrictions:
  - Read-Only source(s) to Read-Only target is allowed
  - Read-Write source(s) to Read-Only target is allowed
  - Read-Only source(s) to Read-Write target is **not** allowed
- Execute database utilities that only perform read operation of the data, for example, BINARY DUMP, TABANALYS.
- Execute database utilities that only change the format of the data, but not the data itself. For example encrypting, decrypting, or changing the code page.
- Execute database utilities that only operate on indexes, for example, IDXBUILD, IDXCOMPACT, IDXCHECK, TPIDXBUILD. Operations within IDXFIX that operate on indexes only are also permitted.
- Execute rescue operations with DBTOOL.
- Restore from a backup (PROREST) and after-imaging roll forward (RFUTIL). A read-only partition can be overwritten by a restore operation. The final state of the partition depends on the information restored from the backup.

Notes

- PARTITIONMANAGE ALTER can be performed when the database is online or offline.

PROUTIL PARTITIONMANAGE COPY qualifier

Partition management option to copy (restore) data into a partitioned table in the destination database from a temporary source database.

Caution: Both the source and target databases must be enabled and configured for table partitioning. The source database must be in restricted mode for PARTITIONMANAGE COPY to execute. See PROUTIL DBRESTRICT qualifier on page 675 for details on restricting access to your source database.

Syntax

```
proutil db-name -C partitionmanage copy
 [ source source-db
  [ partition table-partition-name |
      composite initial table table-name ]
         [recs n] ] |
      clear all
```

Parameters

db-name

Specifies the table partitioned database where you are performing partition management. This is the destination database of the PARTITIONMANAGE COPY operation.

`source` *source-db*

Specifies the source database for the PARTITIONMANAGE COPY operation. The source database must be enabled for table partitioning.

`partition` *table-partition-name*

Specifies the partition name of the data to copy with the PARTITIONMANAGE COPY operation.

`composite initial table` *table-name*

Specifies that the composite initial partition of the table *table-name* is being copied by the PARTITIONMANAGE COPY operation.

`clear all`

Removes all partition copy pending state information from all partitioned objects in *db-name*. The `clear all` option is only used if *db-name* needs to be used with a version of OpenEdge earlier than Release 11.6.0, and a PARTITIONMANAGE COPY operation terminated prior to completion.

`recs` *n*

Specifies the number of records to restore (copy) in a transaction. The default is 100.

PARTITIONMANAGE COPY copies data, local indexes, lobs and updates global indexes for the specified partition in the destination database (*db-name*). The following conditions must be met before PARTITIONMANAGE COPY can execute:

- The source and destination databases must have identical schema.
- The source database must be in partitioncopy restricted mode.
- Indexes (local and global) of the specified partition must be active. If any index is inactive, the utility exits with an error.
- The source and destination databases must be enabled for Table Partitioning.

If only destination (*db-name*) and source (source *source-db*) databases are specified on the command line, PARTITIONMANAGE COPY invokes a menu-driven user interface as shown:

```
Progress OpenEdge Partition Copy utility

Source Database     : temp_source_db
Destination Database: production_db

1. Copy Table instance by Partition
2. Quit

Enter your selection:
```

The user interface can be used to select multiple partitions to copy at once.

Notes

- PARTITIONMANAGE COPY can be performed when the database is online or offline.
- PARTITIONMANAGE COPY will not operate on read-only partitions.

PROUTIL PARTITIONMANAGE MERGE qualifier

Partition management option to merge multiple range or list-range table partitions into a single target partition.

Syntax

```
proutil db-name -C partitionmanage merge table table-name
  partition table-partition-name partition table-partition-name [ partition
table-partition-name ]
 [ useindex index-name ] [ recs numrecs ]
```

Parameters

db-name

Specifies the database where you are performing partition management.

table *table-name*

Specifies the table for the PARTITIONMANAGE MERGE operation.

partition *table-partition-name* [...]

Specifies the partition instances being merged. You must specify at least two partitions, and you can specify a maximum of three partitions.

If you specify two partitions, the first partition is the "start" partition, and the second is the "end" and implicit the "target" partition. All adjacent partitions between the start and end are merged into the end partition.

If you specify three partitions, the first partition is the "start" partition, and the second is the "end", and the third partition is the "target" partition. All adjacent partitions between the start and end are merged into the target partition.

`useindex` *`index-name`*

Specifies the name of the index to use when scanning the table during the merge. The index must be an active local index, specifying a global index is not supported. If not specified, PARTITIONMANAGE MERGE attempts to locate a unique, partition-aligned index.

`recs` *`numrecs`*

The number of records to merge in one transaction, and the number of records to delete in one transaction for the second phase of the merge operation. If not specified, *`numrecs`* defaults to 100.

PARTITIONMANAGE MERGE is used to combine the records from one or more adjacent partitions into one table partition. Use PARTITIONMANAGE MERGE when you need to combine partitions for simplification or storing historical data in one larger partition. For example, if data was partitioned by month, you could merge all the months from a prior year into one partition.

When the merge operation begins, the source partition is inaccessible to the language clients. When a table partition is merged, the records, LOB fields, and local index partitions of the table partition instance are moved to the appropriate target partition, and global indexes are updated.

During the second phase of PARTITIONMANAGE MERGE, the source table partition storage object and partition policy detail record are deleted.

PARTITIONMANAGE MERGE expects the merge source partitions to be adjacent; non-adjacent partitions cannot be merged. After a merge, the target partitions maintains the upper bound of the merged partitions. The target partition spans the range of the merged partitions.

Notes

- PARTITIONMANAGE MERGE does not support composite partitions; PARTITIONMANAGE MERGE supports range and list-range partitions.
- Executing PARTITIONMANAGE MERGE on list partitions is not supported.
- The target partition of PARTITIONMANAGE MERGE must be allocated.
- Partitions that are in any rebuilding state (from another utility) cannot be merged.
- During the execution of PARTITIONMANAGE MERGE, if the target partition has an active index that is inactive in a source partition, the merge creates index entries for the records in the merged target partition. If a source partition index is active, but inactive in the target partition, no index entries are created in the merged target partition.
- If PARTITIONMANAGE MERGE does not successfully complete, re-executing the command completes the merge, from the last committed transaction.
- If *`db-name`* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and PARTITIONMANAGE MERGE exits.

PROUTIL PARTITIONMANAGE RENAME qualifier

Partition management option to rename a partition of a partitioned table.

Syntax

```
proutil db-name -C partitionmanage rename partition table-partition-name
  partition new-table-partition-name
```

Parameters

db-name

Specifies the database where you are performing partition management.

partition *table-partition-name*

Specifies the current name of the existing partition to rename.

partition *new-table-partition-name*

Specifies the new name of the partition. The name must not exist in the database prior to executing this RENAME command.

The *new-table-partition-name* must follow the standard rules for partition naming. The name:

- Must be unique; it cannot already exist in the database
- Must be 32 or fewer characters
- Can only contain the following characters: a-z, A-Z, 0-9 (ASCII) , and the special characters::# $ % & - (dash) _ (underscore)
- Is not blocked from using reserved words (as the database does for table names)

Notes

- You can not rename the composite initial partition of a table.
- You can not rename a partition that is the source of merge operation. PARTITIONMANAGE RENAME checks the object state of the partition, and if it is involved in a merge, the rename is denied.
- PARTITIONMANAGE RENAME can be performed when the database is online or offline.
- If RENAME is executed while the database is online, it is possible that other partition operations (both language clients and utilities) could report the name change during execution. For example, DBANALYS could possibly report both the original and renamed partition names.

PROUTIL PARTITIONMANAGE SPLIT qualifier

Partition management option to split a table partition into multiple target partitions.

Syntax

```
proutil db-name -C partitionmanage split table table-name
  { partition table-partition-name | composite initial }
  [ useindex index-name ] [ recs numrecs ]
```

Parameters

db-name

Specifies the database where you are performing partition management.

table *table-name*

Specifies the source table for the PARTITIONMANAGE SPLIT operation..

partition *table-partition-name*

Specifies the partition instance being split.

composite initial

Specifies a special case for PARTITIONMANAGE SPLIT. "initial" is a reserved term indicating the name of a table's composite partition during data migration.

useindex *index-name*

Specifies the name of the index to use when scanning the table during the split. The index must be a local index. A non-unique index is allowed, but may adversely impact the performance of the PARTITIONMANAGE SPLIT operation.

res *numrecs*

The number of records to split in one transaction. If not specified, *numrecs* defaults to 100.

PARTITIONMANAGE SPLIT is used to redistribute the records in one table partition into different partitions. Use PARTITIONMANAGE SPLIT when a partition is too big, a partition needs to be partitioned into finer granularities, or for data migration when initially partitioning a table.

When a table partition is split, the records, LOB fields, and local index partitions of the table partition instance are moved to the appropriate target partitions, and global indexes are updated.

PARTITIONMANAGE SPLIT relies on the target partitions being previously defined. PARTITIONMANAGE SPLIT will allocate target partitions that are not allocated at the beginning of the split.

Notes

- PARTITIONMANAGE SPLIT supports multiple target partitions.
- PARTITIONMANAGE SPLIT supports composite partitions, range partitions, or sub-partitions that have range at the end. Executing PARTITIONMANAGE SPLIT on list partitions is not supported, unless the source partition is a composite partition..
- If execution of PARTITIONMANAGE SPLIT does not successfully complete, re-executing the command completes the split, from the last committed transaction.
- PARTITIONMANAGE SPLIT is a resource-intensive operation.
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and PARTITIONMANAGE SPLIT exits.

PROUTIL PARTITIONMANAGE TRUNCATE qualifier

Partition management option to truncate a table partition.

Syntax

```
proutil db-name -C partitionmanage truncate table table-name
    { partition table-partition-name | composite initial }
    [ recs numrecs ] [ deallocate ]
```

Parameters

db-name

Specifies the database where you are performing partition management.

table *table-name*

Specifies the table for the PARTITIONMANAGE TRUNCATE operation.

partition *table-partition-name*

Specifies the partition instance being truncated.

composite initial

Specifies that the PARTITIONMANAGE TRUNCATE operation is to be performed into the composite initial partition of the table.

deallocate

Specifies that the partition is in the deallocated state when the truncate operation completes.

recs *numrecs*

When **deallocate** is specified, **recs** governs the number of key entries to delete from global indexes in one transaction. If not specified, *numrecs* defaults to 100.

PARTITIONMANAGE TRUNCATE is used to free the storage space of a partition. You can use PARTITIONMANAGE TRUNCATE to reclaim space when data associated with a partition is no longer valid or needed. The storage space reclaimed is from the table data of the partition, any LOB's associated with the table data of the partition, and the partition aligned indexes.

During the execution of PARTITIONMANAGE TRUNCATE, the only allowed access is by no-lock readers, which may see no data returned as the truncate executes and the partition becomes deallocated.

Schema definitions for a truncated partition still exist, and are marked as "storage space not allocated." New queries against a truncated partition that is also deallocated partition fail.

Notes

- PARTITIONMANAGE TRUNCATE runs against databases that are both online and offline.
- Multiple instances of PARTITIONMANAGE TRUNCATE can run simultaneously provided they operate on different partitions.
- If you use OpenEdge Replication, partition changes are replicated to your replication target database.
- If execution of PARTITIONMANAGE TRUNCATE does not successfully complete, re-executing the command completes the truncate, from the last committed transaction.

- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and PARTITIONMANAGE TRUNCATE exits.

PROUTIL PARTITIONMANAGE VIEW qualifier

Partition management option to view information about table partitions.

Syntax

```
proutil db-name -C partitionmanage view
    [ table table-name
    [ partition table-partition-name | composite initial ] ]
    { list | state | status }
```

Parameters

db-name

Specifies the database where you are performing partition management.

`table` *table-name*

Specifies the table for the PARTITIONMANAGE VIEW operation.

`partition` *table-partition-name*

Specifies the partition instance that PARTITIONMANAGE VIEW is to report on.

`composite initial`

Specifies that PARTITIONMANAGE VIEW is to report on the composite initial partition of the table.

`list`

Specifies that PARTITIONMANAGE VIEW is to list partitions by partition definition order (adjacent definitions not by partition Id).

`state`

Specifies that PARTITIONMANAGE VIEW is to report on the state of a partition. Partitions states are:

- **No state reported (blank)** — The partition has no special characteristics, and is readable and writable.
- **Rebuilding** — The partition is involved in a partition rebuild operation. The rebuilding type indicates the type of rebuild (split, merge, truncate, etc) and if the partition is the source or target.
- **Read Only** — The partition is set to read only.
- **Deallocated** — The partition is deallocated.

`status`

Specifies that PARTITIONMANAGE VIEW is to report on the status of a partition that is in a rebuilding state. The number of rows in each partition is reported and can be used to measure progress of a partition operation such as split or merge.

PARTITIONMANAGE VIEW reports on the following:

- The current state of a partition or partitions
- The progress of a partition utility operation (such as split, merge, and truncate)
- A list of a table's partition definitions, partition by partition in adjacent order

Sample output

Depending on the qualifier chosen, PARTITIONMANAGE VIEW displays different output. The following samples illustrate the different types:

- **List** output for a partition that is partitioned on three columns

```
PROGRESS Partition View


Database: /usr1/docsample/tpdb
Date: Wed Nov 12 11:10:36 2014

Partition List:
--------------------------
Table          Column Partitioning Values
PUB.order
   RenamedOrderPart1:26
                1
                4
                01/01/2015 00:00:00.000
```

- **State** output for allocated partitions in a default state

```
PROGRESS Partition View


Database: /usr1/docsample/tpdb_np
Date: Wed Nov 12 11:20:24 2014


PARTITION STATE
--------------------
Table                                        State                                Description
PUB.history
  initial:0
  hist_ware1_to2020:0
```

- **State** output for a partition in a split-pending state

```
PROGRESS Partition View


Database: /usr1/docsample/tpdb_np
Date: Wed Nov 12 11:30:24 2014

PARTITION STATE
---------------------
Table                           State                                 Description
PUB.history
  initial:0                     Read Only
  hist_ware1_to1980:2           Rebuilding, Deallocated, Read Only    Split (T) Source =
Composite initial:0
  hist_ware1_to1990:3           Rebuilding, Deallocated, Read Only    Split (T) Source =
Composite initial:0
  hist_ware1_to2000:4           Rebuilding, Deallocated, Read Only    Split (T) Source =
Composite initial:0
  hist_ware1_to2010:5           Rebuilding, Deallocated, Read Only    Split (T) Source =
Composite initial:0
  hist_ware1_to2020:1           Rebuilding, Deallocated, Read Only    Split (T) Source =
Composite initial:0
```

- **Status** output for a partition

```
PROGRESS Partition View


Database: /usr1/docsample/tpdb
Date: Wed Nov 12 11:40:51 2014

PARTITION STATUS
----------------------
Table                                                          Rows
PUB.customer
  initial:0                                                    0
  cust_id_llr_10:10                                            0
  cust_id_llr_8:8                                              26
  cust_id_llr_7:7                                              34
  cust_id_llr_9:9                                              28
  cust_id_llr_6:6                                              41
  cust_id_llr_5:5                                              21
  cust_id_llr_3:3                                              35
  cust_id_llr_2:2                                              26
  cust_id_llr_4:4                                              37
  cust_id_llr_1:1                                              1
  cust_id_llr_20:20                                            0
  cust_id_llr_18:18                                            42
  cust_id_llr_17:17                                            64
  cust_id_llr_19:19                                            72
  cust_id_llr_16:16                                            58
  cust_id_llr_15:15                                            67
  cust_id_llr_13:13                                            74
  cust_id_llr_12:12                                            0
  cust_id_llr_14:14                                            62
  RenamedCustomerPart1:11                                      9
  cust_id_llr_30:30                                            0
  cust_id_llr_28:28                                            52
  cust_id_llr_27:27                                            51
  cust_id_llr_29:29                                            58
  cust_id_llr_26:26                                            67
  cust_id_llr_25:25                                            51
  cust_id_llr_23:23                                            51
  cust_id_llr_22:22                                            66
  cust_id_llr_24:24                                            74
  cust_id_llr_21:21                                            4
  cust_id_llr_40:40                                            0
  cust_id_llr_38:38                                            53
  cust_id_llr_37:37                                            53
  cust_id_llr_39:39                                            68
  cust_id_llr_36:36                                            87
  cust_id_llr_35:35                                            66
  cust_id_llr_33:33                                            62
  cust_id_llr_32:32                                            64
  cust_id_llr_34:34                                            53
  cust_id_llr_31:31                                            2
  cust_id_llr_50:50                                            0
  cust_id_llr_48:48                                            75
  cust_id_llr_47:47                                            65
  cust_id_llr_49:49                                            74
  cust_id_llr_46:46                                            74
  cust_id_llr_45:45                                            59
  cust_id_llr_43:43                                            60
  cust_id_llr_42:42                                            50
  cust_id_llr_44:44                                            39
  cust_id_llr_41:41                                            5
```

Note: You can use the number of rows to track the progress of partition operations such as split or merge.

Notes

- PARTITIONMANAGE VIEW runs against databases that are both online and offline.
- Multiple instances of PARTITIONMANAGE VIEW can run simultaneously provided they operate on different partitions.
- PARTITIONMANAGE VIEW runs against both Replication source and target databases.
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and PARTITIONMANAGE VIEW exits.

PROUTIL RCODEKEY qualifier

Inserts the authorization key into existing CRC-based r-code.

Syntax

```
proutil db-name -C rcodekey old-key new-key files . . .
```

Parameters

db-name

Specifies the database you are using.

old-key

Specifies the old authorization key.

new-key

Specifies the new authorization key.

files

Specifies the r-code files.

Notes

- When you use the DBAUTHKEY qualifier to set, change, and remove authorization keys, you can use the RCODEKEY qualifier to update your r-code without compiling it.

  The following table lists the values you must enter for each task:

Task	old-key value	new-key value	Files
Set the authorization key	Plus sign (+)	Authorization key	R-code files
Change the authorization key	Current authorization key	New authorization key	R-code files
Remove the authorization key	Current authorization key	Plus sign (+)	R-code files

- Once you set the authorization key, do not forget it. You cannot change or remove the authorization key without knowing its current value.

PROUTIL SETAREACREATELIMIT qualifier

Set the create limit for the specified area.

Syntax

```
proutil db-name -C setareacreatelimit area-name create-limit
        [ tenant tenant-name | group group-name | shared ]
```

Parameters

db-name

Specifies the database where you are setting the create limit.

area-number

Specifies the area where you are setting the create limit. You can specify an area by its name or its area number.

create-limit

Specifies the new create limit. The create limit is the amount of free space that must remain in a block when you are adding a record to the block.

`tenant` *tenant-name*

Specifies that the area create limit is to be set on tables and LOB objects in the area that are associated with *tenant-name*. If *tenant-name* is not valid, is part of a group, or is not associated with a partition in the area, SETAREACREATELIMIT exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy and Type II areas.

`group` *group-name*

Specifies that the area create limit is to be set on tables and LOB objects in the area that are associated with *group-name*. If *group-name* is not valid or is not associated with partition in the area, SETAREACREATELIMIT exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy and Type II areas.

`shared`

Specifies that the area create limit is to be set on shared tables and LOB objects in the area. Specifying `shared` is only allowed on databases enabled for multi-tenancy and Type II areas.

Notes

- SETAREACREATELIMIT will set the limit for all tables and LOBs in the specified area.
- The create limit value must be equal to or greater than 32, and less than the block size minus 128 bytes.

- For databases with a 1K block size, the default create limit is 75. For all other database block sizes, the default create limit is 150.
- For databases enabled for multi-tenancy, if `tenant`, `group`, or `shared`, is not specified, and the area has multi-tenant objects, the create limit is set for all objects in the area.
- For more information on setting create limits, see Database fragmentation on page 308 .

PROUTIL SETAREATOSSLIMIT qualifier

Set the toss limit for the specified area.

Syntax

```
proutil db-name -C setareatosslimit area-name toss-limit
     [ tenant tenant-name | group group-name | shared ]
```

Parameters

db-name

Specifies the database where you are setting the toss limit.

area-name

Specifies the area where you are setting the toss limit. You can specify an area by its name or its area number.

toss-limit

Specifies the new toss limit. The toss limit is the minimum amount of free space that must exist in a block for the block to remain on the free chain as a candidate for holding additional records.

`tenant` *tenant-name*

Specifies that the area toss limit is to be set on tables and LOB objects in the area that are associated with *tenant-name*. If *tenant-name* is not valid, is part of a group, or is not associated with a partition in the area, SETAREATOSSLIMIT exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy and Type II areas.

`group` *group-name*

Specifies that the area toss limit is to be set on tables and LOB objects in the area that are associated with *group-name*. If *group-name* is not valid or is not associated with partition in the area, SETAREATOSSLIMIT exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy and Type II areas.

`shared`

Specifies that the area toss limit is to be set on shared tables and LOB objects in the area. Specifying `shared` is only allowed on databases enabled for multi-tenancy and Type II areas.

Notes

- SETAREATOSSLIMIT will set the limit for all tables and LOBs in the specified area.
- The toss limit value must be greater than zero (0) and less than the block size minus 128 bytes.
- For databases with a 1K block size, the default toss limit is 150. For all other database block sizes, the default toss limit is 300.
- For databases enabled for multi-tenancy, if `tenant`, `group`, or `shared`, is not specified, and the area has multi-tenant objects, the toss limit is set for all objects in the area.
- For more information on setting toss limits, see Database fragmentation on page 308.

PROUTIL SETLOBCREATELIMIT qualifier

Set the create limit for the specified LOB.

Syntax

```
proutil db-name -C setlobcreatelimit lob-name create-limit
[ tenant tenant-name | group group-name |
          partition partition-name | composite initial]
```

Parameters

db-name

Specifies the database where you are setting the create limit.

lob-name

Specifies the LOB object for which you are setting the create limit.

create-limit

Specifies the new create limit.

`tenant` *tenant-name*

Specifies that the create limit is to be set on the LOB object associated with *tenant-name*. If *tenant-name* is not valid, is part of a group, or is not associated with the specified LOB object, SETLOBCREATELIMIT exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *group-name*

Specifies that the create limit is to be set on LOB object associated with *group-name*. If *group-name* is not valid or is not associated with the specified LOB object, SETLOBCREATELIMIT exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`partition partition-name`

Specifies that the create limit is to be set on LOB object associated with the table partition *partition-name*. If *partition-name* is not valid or is not associated with the specified LOB object, SETLOBCREATELIMIT exits with an error. Specifying `partition` is only allowed on databases enabled for table partitioning.

`composite initial`

Specifies that the create limit is to be set on LOB object associated with the composite initial partition of *lob-name*. If *lob-name* is not associated with a partitioned table, SETLOBCREATELIMIT exits with an error. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

Notes

- For a Type II area, SETLOBCREATELIMIT sets the LOB create limit for the specified LOB. For a Type I area, SETLOBCREATELIMIT issues a warning that all tables and LOBs in an area must have the same create limit, and prompts you to confirm setting the create limit for all tables and LOBs in the area containing the specified LOB.
- The create limit value must be equal to or greater than 32, and less than the block size minus 128 bytes.
- For databases with a 1K block size, the default create limit is 75. For all other database block sizes, the default create limit is 150.
- For more information on setting create limits, see Database fragmentation on page 308.

PROUTIL SETLOBTOSSLIMIT qualifier

Set the toss limit for the specified LOB.

Syntax

```
proutil db-name -C setlobtosslimit lob-name toss-limit
    [ tenant tenant-name | group group-name |
                partition partition-name | composite initial ]
```

Parameters

db-name

Specifies the database where you are setting the toss limit.

lob-name

Specifies the LOB object for which you are setting the toss limit.

toss-limit

Specifies the new toss limit.

`tenant` *`tenant-name`*

Specifies that the toss limit is to be set on the LOB object associated with *`tenant-name`*. If *`tenant-name`* is not valid, is part of a group, or is not associated with the specified LOB object, SETLOBTOSSLIMIT exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *`group-name`*

Specifies that the toss limit is to be set on LOB object associated with *`group-name`*. If *`group-name`* is not valid or is not associated with partition in the area, SETLOBTOSSLIMIT exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`partition` *`partition-name`*

Specifies that the toss limit is to be set on LOB object associated with the table partition *`partition-name`*. If *`partition-name`* is not valid or is not associated with the specified LOB object, SETLOBTOSSLIMIT exits with an error. Specifying `partition` is only allowed on databases enabled for table partitioning.

`composite initial`

Specifies that the toss limit is to be set on LOB object associated with the composite initial partition of *`lob-name`*. If *`lob-name`* is not associated with a partitioned table, SETLOBTOSSLIMIT exits with an error. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

Notes

- For a Type II area, SETLOBTOSSLIMIT sets the LOB toss limit for the specified LOB. For a Type I area, SETLOBTOSSLIMIT issues a warning that all tables and LOBs in an area must have the same toss limit, and prompts you to confirm setting the toss limit for all tables and LOBs in the area containing the specified LOB.
- The toss limit value must be greater than zero (0) and less than the block size minus 128 bytes.
- For databases with a 1K block size, the default toss limit is 150. For all other database block sizes, the default toss limit is 300.
- For more information on setting toss limits, see Database fragmentation on page 308.

PROUTIL SETTABLECREATELIMIT qualifier

Set the create limit for the specified table.

Syntax

```
proutil db-name -C settablecreatelimit table-name create-limit
 [ tenant tenant-name | group group-name |
           partition partition-name | composite initial ]
```

Parameters

db-name

Specifies the database where you are setting the create limit.

table-name

Specifies the table where you are setting the create limit.

create-limit

Specifies the new create limit. The create limit is the amount of free space that must remain in a block when you are adding a record to the block.

tenant *tenant-name*

Specifies that the table create limit is to be set on the table partition that is associated with *tenant-name*. If *tenant-name* is not valid, is part of a group, or is not associated with a partition of the table, SETTABLECREATELIMIT exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the table create limit is to be set on the table partition that is associated with *group-name*. If *group-name* is not valid or is not associated with a partition of the table, SETTABLECREATELIMIT exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

partition *partition-name*

Specifies that the table create limit is to be set on the table partition *partition-name*. If *partition-name* is not valid or is not associated with the specified table, SETABLECREATELIMIT exits with an error. Specifying `partition` is only allowed on databases enabled for table partitioning.

composite initial

Specifies that the table create limit is to be set on the composite initial partition of the specified table. If *table-name* is not a partitioned table, SETTABLECREATELIMIT exits with an error. Specifying `composite initial` is only allowed on databases enabled for table partitioning.

Notes

- For a Type II area, SETTABLECREATELIMIT sets the table create limit for the specified table. For a Type I area, SETTABLECREATELIMIT issues a warning that all tables and LOBs in an area must have the same create limit, and prompts you to confirm setting the create limit for all tables and LOBs in the area containing the specified table.
- The create limit value must be equal to or greater than 32, and less than the block size minus 128 bytes.
- For databases with a 1K block size, the default create limit is 75. For all other database block sizes, the default create limit is 150.
- For databases enabled for table partitioning, if the specified *table-name* is a partitioned table, but `partition` or `composite initial` is not specified, then the create limit is set for all partitions of the table.
- For more information on setting create limits, see Database fragmentation on page 308.

PROUTIL SETTABLETOSSLIMIT qualifier

Set the toss limit for the specified table.

Syntax

```
proutil db-name -C settabletosslimit table-name toss-limit
   [ tenant tenant-name | group group-name |
              partition partition-name | composite initial ]
```

Parameters

db-name

Specifies the database where you are setting the toss limit.

table-name

Specifies the table where you are setting the toss limit.

toss-limit

Specifies the new toss limit.

tenant *tenant-name*

Specifies that the table toss limit is to be set on the table partition that is associated with *tenant-name*. If *tenant-name* is not valid, is part of a group, or is not associated with a partition of the table, SETTABLETOSSLIMIT exits with an error. Specifying tenant is only allowed on databases enabled for multi-tenancy.

group *group-name*

Specifies that the table toss limit is to be set on the table partition that is associated with *group-name*. If *group-name* is not valid or is not associated with a partition of the table, SETTABLETOSSLIMIT exits with an error. Specifying group is only allowed on databases enabled for multi-tenancy.

partition *partition-name*

Specifies that the table toss limit is to be set on the table partition *partition-name*. If *partition-name* is not valid or is not associated with the specified table, SETABLETOSSLIMIT exits with an error. Specifying partition is only allowed on databases enabled for table partitioning.

composite initial

Specifies that the table toss limit is to be set on the composite initial partition of the specified table. If *table-name* is not a partitioned table, SETTABLETOSSLIMIT exits with an error. Specifying composite initial is only allowed on databases enabled for table partitioning.

Notes

- For a Type II area, SETTABLETOSSLIMIT sets the table toss limit for the specified table. For a Type I area, SETTABLETOSSLIMIT issues a warning that all tables and LOBs in an area must have the same toss limit, and prompts you to confirm setting the toss limit for all tables and LOBs in the area containing the specified table.
- The toss limit value must be greater than zero (0) and less than the block size minus 128 bytes.
- For databases with a 1K block size, the default create limit is 150. For all other database block sizes, the default create limit is 300.
- For databases enabled for table partitioning, if the specified *table-name* is a partitioned table, but `partition` or `composite initial` is not specified, then the toss limit is set for all partitions of the table.
- For more information on setting toss limits, see Database fragmentation on page 308.

PROUTIL TABANALYS qualifier

Displays information about the degree of fragmentation for each table in a database. Also displays summary information about record and LOB (if present) sizes for each table.

Syntax

```
proutil db-name -C tabanalys [ area area-name ]
     [ tenant tenant-name | group group-name | shared ]
     [-csoutput [ -fieldsep sep-value] [ -csfilePrefix pref-name ]
          [ -verbose ] ]
```

Parameters

db-name

Specifies the database to display information.

`area` *area-name*

Specifies that TABANALYS is to be performed on the *area-name* area only.

`tenant` *tenant-name*

Specifies that TABANALYS is to be filtered to only provide information related to objects that are owned by *tenant-name*. If *tenant-name* is not valid, TABANALYS exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *group-name*

Specifies that TABANALYS is to be filtered to only provide information related to objects that are owned by *group-name*. If *group-name* is not valid, TABANALYS exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`shared`

Specifies that TABANALYS is to be filtered to only provide information related to shared objects. Specifying `shared` is only allowed on databases enabled for multi-tenancy.

`-csoutput`

Specifies that TABANALYS output the analysis data to text files in addition to the screen. The result is that the following files are created:

- *db-name*`.tab.txt` — For table analysis output
- *db-name* `.block.txt` — For block analysis output
- *db-name* `.lob.txt` — For LOB analysis output

For more information about the contents of the files, see the Database analysis output on page 667.

`-fieldsep` *sep-value*

When `-csoutput` is specified, *sep-value* indicates the value to separate the columns of output. A space is used by default. Accepted values for *sep-value* are:

- Any ASCII character between 0x21 and 0x7E
- `sp` — to denote a space (the default)
- `tab` — to denote a the TAB character (`<\t>` or ASCII character 0x09)

`-csfilePrefix` *pref-name*

When `-csoutput` is specified, *pref-name* indicates a file name prefix. The database name is used by default. All remaining elements of the file name are not customizable.

`-verbose`

When `-csoutput` is specified, the first row of the output files contains header names for each column of output.

Output format

In a complete report where no optional elements are specified on the command line, shared objects (both partitioned and non-partitioned) appear in alphabetical order by name. Partitions of partitioned tables are ordered as follows: the initial partition, then the remainder of the partitions in alphabetical order by partition name.

Example

The following output is a sample display of PROUTIL TABANALYS:

```
RECORD BLOCK SUMMARY FOR AREA "DataArea": 17
------------------------------------------------------

RECORD BLOCK SUMMARY FOR SHARED TABLES
--------------------------------------

                                    -Record Size (B)- ---Fragments--- Scatter
Table                 Records    Size   Min   Max  Mean  Count Factor Factor
PUB.cust                   26    1.7K    55    76    68 26    1.0     1.0
PUB.cust-list.Initial       0    0.0B     0     0     0      0    0.0     0.0
PUB.cust-list.list1         3  202.0B    66    68    67      3    1.0    20.3
```

```
PUB.cust-list.list2        10  645.0B    56    77    64     10    1.0    20.3
PUB.cust-list.list3         4  291.0B    71    74    72      4    1.0    20.3
PUB.cust-list-range.Initial 0    0.0B     0     0     0      0    0.0     0.0
PUB.cust-range.Initial      0    0.0B     0     0     0      0    0.0     0.0
PUB.cust-rvalue.Initial     0    0.0B     0     0     0      0    0.0     0.0
PUB.state                  51    1.7K    29    40    34     51    1.0     1.0
PUB.stock.Initial(Composite)
                        50000   18.5M   368   406   387  50000    1.0     1.0
.
.
.
LOB SUMMARY FOR AREA dbArea3: 9


LOB SUMMARY FOR SHARED TABLES:
----------------------------------------------


                                                        ---    LOB Size   ---
Table                                  LOBs     Size      Min      Max     Mean
PUB.Invoice:1
    InvBlob1:5                          400    55.1M     1.0B   976.6K   141.0K
    InvBlob2:6                          400    54.9M     1.0B   976.6K   140.4K
    InvClob1:7                          400    39.3M     1.0B   976.6K   100.5K
                                  ----------------------------------------------
Subtotals:                             1200   149.2M     1.0B   976.6K   127.3K

Summary for AREA "dbArea3": 9
                                  ----------------------------------------------
Subtotals:                             1200   149.2M     1.0B   976.6K   127.3K

.
.
.
.
```

`Table`

Table owner and table name.

`Records`

Total number of records in the database for the table.

`Bytes`

Total number of bytes used in the database for the table.

`Min`

Minimum number of bytes used by any record for the table.

`Max`

Maximum number of bytes used by any record for the table.

`Mean`

Mean number of bytes used by any record for the table.

Count

Total number of record fragments found in the database for the table.

Factor

Degree of record fragmentation for the table. This value is determined by the number of fragments divided by the ideal number of fragments (for example, if the data is freshly loaded). A value of 1.0 is ideal. A value of 2.0 indicates that there are twice as many fragments than there would be when the data is freshly loaded.

Use the Index value to determine whether to dump and reload your data to improve fragmentation. If the value is 2.0 or greater, dumping and reloading will improve performance. If the value is less than 1.5, dumping and reloading is not warranted.

Scatter Factor

Degree of distance between records in the table.

The best achievable Scatter Index value is that of a freshly loaded database. This is the baseline number against which you should compare future Scatter Index measurements. A value of 1 indicates that the records occupy completely contiguous database blocks. A value of 2 indicates that the records are scattered ten times wider than the ideal.

Use this value to determine the performance impact caused by fragmentation. If the value is 1.5 or greater, performance will be poor for sequential record access, and dumping and loading the data might be warranted. A value of 1.5 or greater might also reduce performance for random access; however, this depends on the system type and the application.

Notes

- The PROUTIL DBANALYS display includes the information displayed by PROUTIL TABANALYS.
- If you specify a specific area for analysis, and the area contains LOB data, but not the corresponding records, a warning message is issued indicating that totals cannot be calculated for the LOBs in the LOB Summary and Database Summary sections of the output.
- See Managing Performance on page 275 for more information about database fragmentation.

PROUTIL TABLEMOVE qualifier

Moves a table, or a partition of a table (partitioned or multi-tenant), and optionally moves its associated indexes from one storage area to another while the database remains online.

Syntax

```
proutil db-name -C tablemove
     [owner-name.] table-name table-area [index-area]
     [ tenant tenant-name | group group-name |
        partition partition-name | composite initial]
```

Parameters

db-name

Specifies the name of the database containing the table.

owner-name

Specifies the owner of the table containing the data you want to move. You must specify an owner name unless the table's name is unique within the database, or the table is owned by PUB. By default, ABL tables are owned by PUB.

table-name

Specifies the name of the table to be moved.

table-area

Specifies the area name of the target application data area into which the table is to be moved. Area names with spaces in the name must be quoted, for example, "Area Name."

index-area

Optionally, specifies the name of the target index area. If the target index area is supplied, the indexes will be moved to that area. Otherwise they will be left in their existing location. You can move indexes to an area other than the area to which the table is being moved. Area names with spaces in the name must be quoted, for example, "Area Name."

For a table-partitioned table, only local indexes of a specified partition are moved.

tenant *tenant-name*

For databases enabled for multi-tenancy, specifies that the partition of the table owned by *tenant-name* is to be moved.

If tenant is specified, and *tenant-name* is not valid, or the table is not a multi-tenant table, TABLEMOVE exits with an error.

group *group-name*

For databases enabled for multi-tenancy, specifies that the partition of the table owned by *group-name* is to be moved.

If group is specified, and *group-name* is not valid, or the table is not a multi-tenant table, TABLEMOVE exits with an error.

partition *partition-name*

For databases enabled for table partitioning, specifies that the partition named *partition-name* is to be moved.

If partition is specified, and *partition-name* is not a valid partition, or the table is not a partitioned table, TABLEMOVE exits with an error.

composite initial

For databases enabled for table partitioning, specifies that the initial composite partition is to be moved.

Notes

- If you omit the *index-area* parameter the indexes associated with the table will not be moved.
- Moving the records of a table from one area to another invalidates all the ROWIDs and indexes of the table. Therefore, the indexes are rebuilt automatically by the utility whether you move them or not. You can move the indexes to an application data area other than the one to which you are moving the table. If you want to move only the indexes of a table to a separate application data area, use the PROUTIL IDXMOVE utility.

  Moving a table's indexes with the TABLEMOVE qualifier is more efficient than moving a table separately and then moving the indexes with the IDXMOVE utility. Moving a table separately from its indexes wastes more disk space and causes the indexes to be rebuilt twice, which also takes longer.

  The PROUTIL TABLEMOVE utility operates in four phases:

  - **Phase 1** — The records are moved to the new area and a new primary key is built.
  - **Phase 2** — All the secondary indexes are built.

    If you did not specify the *index-area* parameter, then the indexes are rebuilt in their original area. If you did specify the *index-area* parameter, then all the indexes are moved to the new area where they are rebuilt.
  - **Phase 3** — All the records in the old area are removed.
  - **Phase 4** — All the old indexes are killed and the _StorageObject records of the indexes and the table are updated.
- Although PROUTIL TABLEMOVE operates in phases, it moves a table and its indexes in a single transaction. To allow a full recovery to occur when a transaction is interrupted, every move and delete of each individual record is logged. As a result, moving a table requires the BI Recovery Area to be several times larger than the combined size of the table and its indexes. Therefore, before moving your table, determine if your available disk capacity is sufficient to support a variable BI extent that might grow to more than three times the size of the table and its indexes.
- While you can move tables online, no access to the table or its indexes is recommended during the move. The utility acquires an EXCLUSIVE lock on the table while it is moving. An application that reads the table with an explicit NO-LOCK might be able to read records, but in some cases might get the wrong results, since the table move operation makes many changes to the indexes. Run the utility during a period when the system is relatively idle, or when users are doing work that does not access the table.
- LOBs are **not** moved during the TABLEMOVE operation.
- No other administrative operation on any index of the moved table is allowed during the table move.
- There is a possibility that the utility will have to wait for all the necessary locks to be available before it can start, which may take some time.
- The `_UserStatus` virtual system table displays the utility's progress. See Maintaining Database Structure on page 325 for more information.
- For multi-tenant tables, you can not specify both `group` and `tenant`, but you must specify one. The *table-area* and *index-area* parameters must specify Type II areas.
- Shared tables of a multi-tenant database are moved without specifying `tenant` or `group` on the command line.
- For table-partitioned tables, you can not specify both `partition` and `composite initial`. The *table-area* and *index-area* parameters must specify Type II areas.
- Shared tables of a table-partitioned database are moved without specifying `partition` or `composite initial` on the command line.

PROUTIL TPIDXBUILD qualifier

For databases enabled for table partitioning, rebuild an index or indexes for a single partition of a partitioned table, packing or consolidating index records to use disk space as efficiently as possible.

Syntax

```
proutil db-name -C tpidxbuild table [owner-name.] table-name
   [ area area-name | index index-name ]
   { partition partition-name | composite initial }
   [ -T dir-name | -SS sort-file-directory-specification ]
   [ -TB blocksize ] [ -TM n ] [ -B n] [ -SG n ]
  [ -pfactor n ] [ refresh n ]
```

Parameters

db-name

Specifies the table-partitioned database you are using.

table [*owner-name.*] *table-name*

Specifies the table for the index or indexes to be rebuilt.

area *area-name*

Specifies that you want to rebuild the local indexes for the *partition-name* partition of *table-name*, defined in the named area.

index *index-name*

Specifies one named index for the *partition-name* partition of *table-name* to be rebuilt. The index must be a local index.

partition *partition-name*

Specifies that the TPIDXBUILD operation is to be performed only on the indexes of the *partition-name* table partition. If *partition-name* is not a valid partition for the specified table, or if the partition is in a rebuilding state, TPIDXBUILD exits with an error.

composite initial

Specifies that the TPIDXBUILD operation is to be performed only on the composite initial partition of the table. If the partition is in a rebuilding state, TPIDXBUILD exits with an error.

-T *dir-name*

Specifies the name of the directory where the temporary files are stored. If you do not use this parameter, TPIDXBUILD places temporary files in the current working directory.

-SS *sort-file-directory-specification*

Identifies the location of a multi-volume sort file specification. If you use the Sort Directory Specification (-SS) parameter, TPIDXBUILD does not use the Temporary Directory (-T) parameter.

-TB *n*

Specifies that the table-partition index rebuild will be performed using Speed Sort. *n* indicates the allocated block size, in kilobytes. The maximum value is 64; if not specified, the default is 8.

-TM *n*

Specifies the merge number. *n* indicates the number of blocks or streams to be merged during the sort process.

-B *n*

Specifies the number of blocks in the database buffers.

-SG *n*

Specifies that the table-partition index rebuild uses index grouping. *n* must be a value between 8 and 64, and indicates the number of index groups used by TPIDXBUILD. The default value is 48. Note that a temporary file is created for each index group.

A large -SG value requires more memory allocation and more file handles. To determine the amount of memory (in kilobytes) needed for each index group, add 1 to the merge number (the value of -TM) and multiply the sum by the speed sort block size (the value of -TB). Memory consumption for each index group equals (-TM + 1) * -TB.

-pfactor *n*

Specifies that TPIDXBUILD will use a packing factor. *n* must be a value between 60 and 100, and indicates the maximum percentage of space used in an index block. The default value is 100.

refresh *n*

Specifies the frequency in seconds to update the display of clients that are blocking the build of an index. The default refresh rate is 60 seconds.You can set it as high as 300 seconds. Connected clients with a schema timestamp earlier than the index's schema timestamp will prevent TPIDXBUILD from building the index until they are disconnected.

Notes

- If you do not specify an area or a specific index, all local indexes associated with the table, for the specified partition, are rebuilt. Global indexes are skipped without error.
- You can run multiple instances of TPIDXBUILD at the same time.
- To avoid name collisions with the .srt file when running multiple instances of TPIDXBUILD, the naming convention used for the .srt file is *databasename.partitionname*.srt. If a file with that name is not located, TPIDXBUILD then looks for a file named *databasename*.srt.
- Index partitions that are not allocated can not be rebuilt.
- Use the Temporary Directory (-T) startup parameter to identify or redirect temporary files created by TPIDXBUILD to a specified directory when sorting and handling space issues.
- Use the Speed Sort (-TB), Merge Number (-TM), Sort Grouping (-SG) and Blocks in Database Buffers (-B) startup parameters to improve index rebuild performance.

- TPIDXBUILD prompts you to confirm you have enough disk space for sorting. If you answer “no”, TPIDXBUILD executes without sorting.
- TPIDXBUILD does not repair corrupted record data.
- To determine whether you have enough free space to sort the indexes, consider that when rebuilding an individual index, sorting that index requires free space that can be as large as three times the size of one index entry times the number of records in the file.
- During the table-partitioned index rebuild, the `_index` records for the indexes to be rebuilt are first changed to “active”, and the `_storageObject` record for the partition of the partitioned table, is set to “inactive”. This enables an index to be in different states for different partitions.
- If auditing is enabled on your database:
  - If there is an audit policy defined to record updates on `_index` records for the indexes to be rebuilt, auditing does not record updates.
  - TPDIXBUILD uses the same audit event as IDXBUILD.
- To prevent users from accessing a rebuilt index with a stale cursor, all ABL clients, with a timestamp older than a current timestamp of TPIDXBUILD, must update their cache or log out.

  If TPIDXBUILD detects one or more these clients, there are two possible outcomes:
  - For OpenEdge Release 11.7 and later, if you have enabled Database Client Notification (`-usernotifytime`), TPIDXBUILD waits until the clients respond to the notification, and then proceeds.
  - For releases prior to Release 11.7, or if you have not enabled Database Client Notification (`-usernotifytime 0`), you may wait for those clients to disconnect or forcibly disconnect them with PROSHUT.
- TPIDXBUILD acquires an exclusive table partition lock. Once the build begins, all other processes are blocked from creating or deleting index objects, and clients cannot use the index in any query (locked or no-lock).
- If *db-name* is a UTF-8 database, you must specify `-cpinternal UTF-8` on the command line. If not specified, an error message is generated and TPIDXBUILD exits.

PROUTIL TRUNCATE AREA qualifier

Deletes all data from the tables and indexes in the specified storage area.

Syntax

```
proutil db-name -C truncate area area-name
                [-userid username [-password passwd ] ]
```

Parameters

db-name

Specifies the database that contains the application data storage areas that you want to truncate.

area-name

Specifies the name of the storage area you want to truncate. When you specify the area name, PROUTIL truncates the area even if it contains storage objects. If no area name is specified, PROUTIL truncates all areas not containing objects.

-userid *username*

Identifies a user privileged to access protected audit data when executing this utility.

-password *passwd*

Password for the privileged user.

PROUTIL TRUNCATE AREA works by resetting the high-water mark in the storage area back to the beginning of the storage area. This hi-water mark reset frees all of the space in the storage area for reuse. Any tables and indexes in the storage areas are initialized to the state they were in before they contained any rows or index entries. Before resetting the hi-water mark, the before image (BI) file is truncated.

Notes

- Use of this qualifier is an important step in removing application data storage areas and extents from a database.
- Deleting the contents of storage areas with this feature also allows for rapid dumping and loading. Use PROUTIL TRUNCATE AREA after dumping data, but before initiating the load.
- TRUNCATE AREA requires additional security when auditing is enabled for your database. Depending on how user identification is established for your database, you may need to specify the -userid and -password parameters to run this utility. For more information on auditing and utility security, see Auditing impact on database utilities on page 193.
- To use this command, the database must be offline and after-imaging must be disabled.
- If the storage area does not contain any storage objects, then the command simply resets the hi-water mark. If the storage area does contain tables and or indexes, their names are listed and you must confirm to truncate the storage area.
- Indexes in other storage areas that are on tables in the storage area being truncated are marked as inactive.
- Empty index root blocks for indexes in the area being truncated are recreated.
- PROUTIL TRUNCATE AREA recreates any template records in the new area.
- See Maintaining Database Structure on page 325 for an example of PROUTIL TRUNCATE AREA.

PROUTIL TRUNCATE BI qualifier

Performs the following three functions:

- Uses the information in the before-image (BI) files to bring the database and after-image (AI) files up to date, waits to verify that the information has been successfully written to the disk, then truncates the before-image file to its original length.
- Sets the BI cluster size using the Before-image Cluster Size (-bi) parameter.
- Sets the BI block size using the Before-image Block Size (-biblocksize) parameter.

Syntax

```
proutil db-name -C truncate bi { [ -G n] | -bi size | -biblocksize size }
```

Parameters

db-name

Specifies the database you are using.

-G *n*

Specifies the number of seconds the TRUNCATE BI qualifier waits after bringing the database and AI files up to date and before truncating the BI file. The default wait period is 60 seconds. You might specify a shorter period for practice or test purposes only.

Caution: Do not decrease the value for any significant database, because a system crash could damage the database if the BI file is truncated before the writes to the database and AI files are flushed from the operating system buffer cache.

-bi *size*

Specifies the size of the cluster in kilobytes. The number must be a multiple of 16 ranging from 16 to 262,128 (16K to 256MB). The default cluster size is 512K. If you use a value that is not a multiple of 16, PROUTIL rounds the value up to the next multiple of 16.

-biblocksize *size*

Specifies the size of the BI blocks in each buffer in kilobytes. The valid values are 1, 2, 4, 8, and 16. The default -biblocksize is 8K. A value of zero (0) tells PROUTIL to use the default block size. The block size cannot be smaller than the database block size.

Notes

- The Before-image Block Size (-biblocksize) parameter changes the BI block size so that the database engine reads and writes the blocks as one block.
- Use the PROSTRCT STATISTICS qualifier to display the block size for a database.
- If you change the BI block size or cluster size before backing up a database, when you restore the database, the blocks and clusters will be the size you specified before the backup.
- PROUTIL reads all the BI blocks according to the size of the first block it reads. For example, if the first BI block is 8K, PROUTIL reads and writes each block on an 8K boundary.
- For performance reasons, you might want to run PROUTIL BIGROW to increase the number of BI clusters available to your database before starting your server.

PROUTIL UPDATESCHEMA qualifier

Loads the specified database with newest meta-schema for the current release.

Syntax

```
proutil db-name -C updateschema
```

Parameters

db-name

Specifies the database you are updating.

Notes

- PROUTIL UPDATESCHEMA updates an existing database from a prior release with the newest meta-schema for the current release. All new tables, fields, and indexes are added.
- PROUTIL UPDATESCHEMA updates auditing system events.
- Running PROUTIL UPDATESCHEMA on a database from the current release informs you that your database is up to date.

PROUTIL UPDATEVST qualifier

Loads the specified database with updated or new virtual system tables (VSTs).

Syntax

```
proutil db-name -C updatevst
```

Parameters

db-name

Specifies the database you are using.

Notes

- The PROUTIL utility with UPDATEVST qualifier deletes all existing VST schema information, then recreates all the VSTs from the most current information.
- See Virtual System Tables on page 863for a list of the virtual system tables that OpenEdge provides.

PROUTIL VIEWB2 qualifier

Lists the assignment of areas and objects with respect to alternate buffer pool processing.

Syntax

```
proutil db-name -C viewB2 [ tenant tenant-name | group group-name | shared ]
```

Parameters

db-name

Name of the database where you are viewing alternate buffer pool assignments.

`tenant` *tenant-name*

Specifies that the only the area and object assignments owned by *tenant-name* are to be viewed. If *tenant-name* is not valid, VIEWB2 exits with an error. Specifying `tenant` is only allowed on databases enabled for multi-tenancy.

`group` *group-name*

Specifies that the only the area and object assignments owned by *group-name* are to be viewed. If *group-name* is not valid, VIEWB2 exits with an error. Specifying `group` is only allowed on databases enabled for multi-tenancy.

`shared`

Specifies that the only the area and object assignments of shared tables and objects are to be viewed. Specifying `shared` is only allowed on databases enabled for multi-tenancy.

VIEWB2 displays the alternate buffer pool assignments for objects of all schemas as well as area level assignments. Since objects in a Type II area can be assigned to an alternate buffer pool independently of their area assignment, each object in a Type II area is displayed regardless of the area status. Although objects in a Type I area cannot be explicitly assigned to the alternate buffer pool, they are individually listed to clearly identify the objects associated with a storage area when a storage area is assigned to the alternate buffer pool.

Sample PROUTIL VIEWB2 output

A sample of the output from PROUTIL VIEWB2, follows:

```
Area "Customer/Order Area":8 - Primary Buffer Pool
Object Enablement   Type   Object Name                    Partition/Tenant/Group
-----------------  ------  -----------                    ----------------------
Alternate           Table  PUB.Customer:2                       <none>
Default             Table  PUB.Order:4                          <none>
Default             Table  PUB.Order-Line:5                     <none>
Alternate           Index  PUB.Customer.Sales-Rep:16            <none>
Default             Index  PUB.Order.Cust-Order:21              <none>
Area "Primary Index Area":9 - Primary Buffer Pool

Object Enablement   Type   Object Name                    Partition/Tenant/Group
-----------------  ------  -----------                    ----------------------
Alternate           Index  PUB.Customer.Cust-Num:12             <none>
Default             Index  PUB.Order.Order-Num:20               <none>
Default             Index  PUB.Order-Line.order-line:24         <none>
Area "Customer Index Area":10 - Primary Buffer Pool

Object Enablement   Type   Object Name                    Partition/Tenant/Group
-----------------  ------  -----------                    ----------------------
Alternate           Index  PUB.Customer.Comments:13             <none>
Alternate           Index  PUB.Customer.Country-Post:14         <none>
Alternate           Index  PUB.Customer.Name:15                 <none>
Area "Order Index Area":11 - Primary Buffer Pool

Object Enablement   Type   Object Name                    Partition/Tenant/Group
-----------------  ------  -----------                    ----------------------
Default             Index  PUB.Order.Order-Date:22              <none>
Default             Index  PUB.Order.Sales-Rep:23               <none>
Default             Index  PUB.Order-Line.item-num:25           <none>
```

Notes

- VIEWB2 maybe run online or offline.
- For the `Partition/Tenant/Group` column:
  - Entries preceded by `P:` indicate a table partitioned object.
  - Entries preceded by `T:` indicate an object associated with a multi-tenant tenant.
  - Entries preceded by `G:` indicate an object associated with a multi-tenant group.
  - Index Objects with the form `P:Global:0` are global indexes.
  - Index or Table entries with the form `P:Transitional:0` indicate the composite initial partition.
- Alternate buffer pool assignments for objects visible through ABL (in the PUB schema) can be updated and viewed through the Data Dictionary.
- For more information on assigning objects to the alternate buffer pool, see PROUTIL ENABLEB2 qualifier on page 697 or the Data Dictionary.
- For more information on removing objects from the alternate buffer pool, see PROUTIL DISABLEB2 qualifier on page 683 or the Data Dictionary.

PROUTIL WBREAK-COMPILER qualifier

Compiles a word-break table. For PROUTIL WBREAK-COMPILER to succeed, the word-break table source file must define a data structure named `word_attr`.

Syntax

```
proutil -C wbreak-compiler src-file rule-num
```

Parameters

src-file

Identifies the word-break table source file to compile.

rule-num

Specifies an integer between 1 and 255 (inclusive) that will uniquely identify the compiled word-break table. PROUTIL WORD-BREAK COMPILER names the compiled version of the word-break table `proword.`*rule-num*. For example, if *rule-num* is 34, the name of the compiled word-break table file is `proword.34`.

Notes

- To reference your compiled word-break table (`proword.`*rule-num*), move it to the OpenEdge install directory or set the PROWD environment variable to point to the location of the file. When you set the PROWD environment variable, append the value of *rule-num* to PROWD. For example:

```
PROWD34=proword.34; export PROWD34
```

- To apply the word-break rules to a database, use the WORD-RULES qualifier with the PROUTIL utility.

PROUTIL WORD-RULES qualifier

Compiles a word-rules file.

Syntax

```
proutil db-name -C word-rules rule-numb
```

Parameters

db-name

Specifies the database you are using.

rule-numb

Specifies an integer between 1 and 255 that uniquely identifies this set of rules on your system that you created with the WBREAK-COMPILER qualifier.

Notes

- If you change the word-break rules when word indexes are active, the indexes might not work properly because the rules used to create the index differ from those used when searching the index. Therefore, when you change the break rules for a database, you are warned if any word indexes are active. You should rebuild any such indexes. You can make old indexes consistent with the new rules by rebuilding them with PROUTIL IDXBUILD. See Managing Performance on page 275 for more information about using the IDXBUILD qualifier.
- PROUTIL maintains a CRC code to ensure that the word-break rule file does not change between sessions. If it has changed, PROUTIL displays a message when you attempt to connect to the database. The connect attempt fails. You can fix this by restoring the original file or resetting the break rules to the default. Note that resetting to the default break rules might invalidate your word indexes.

PROUTIL ZEROSTATS qualifier

Resets (zeros) runtime statistics for the specified database.

Syntax

```
proutil db-name -C zerostats
```

Parameters

db-name

Specifies the database where you want to reset the statistics.

PROUTIL ZEROSTATS resets all runtime statistics for the database, including I/O statistics and latch and lock information.

Example

The statistics provided by PROUTIL IOSTATS include buffered, unbuffered, and logical I/O database operations. The statistics are cumulative from database startup or from when PROUTIL ZEROSTATS is executed. For more information, see PROUTIL IOSTATS qualifier on page 738.

The following output is a sample display of PROUTIL IOSTATS:

```
Database name is iostat_ex (2524)
FILE                    BUFFERED         UNBUFFERED         LOGICAL (2556)
                   Writes  Reads   Writes  Reads    Reads  Writes  Extends (2557)
)
================================================================================

iostat_ex.b1            0      0      478     23       21     307        3
iostat_ex.d1            3    236       12      0      234      14        0
iostat_ex_7.d1          1      2        0      0        0       0        0
iostat_ex_7.d2          1      1        0      0        0       0        0
iostat_ex_8.d1          1      2        0      0        0       0        0
iostat_ex_8.d2          1      1        0      0        0       0        0
iostat_ex_9.d1         41      5        0      0        3      40        0
iostat_ex_9.d2         11      1       16      0        0      10        1
iostat_ex_10.d1        25     10        0      0        8      24        0
iostat_ex_10.d2         1      1        0      0        0       0        0
iostat_ex_11.d1         1      2        0      0        0       0        0
iostat_ex_11.d2         1      1        0      0        0       0        0
iostat_ex_12.d1         1      2        0      0        0       0        0
iostat_ex_12.d2         1      1        0      0        0       0        0
iostat_ex.a1            0      0      131      4        3     130        0
iostat_ex.a2            0      0      132      3        2     131        0
iostat_ex.a3            0      0        6      2        1       5        0
```

After running PROUTIL ZEROSTATS on the database, the output of PROUTIL IOSTATS appears as follows:

```
Database name is iostat_ex (2524)
FILE                    BUFFERED         UNBUFFERED         LOGICAL (2556)
                   Writes  Reads   Writes  Reads    Reads  Writes  Extends (2557)
)
================================================================================

iostat_ex.b1            0      0        0      0        0       0        0
iostat_ex.d1            0      0        0      0        0       0        0
iostat_ex_7.d1          0      0        0      0        0       0        0
iostat_ex_7.d2          0      0        0      0        0       0        0
iostat_ex_8.d1          0      0        0      0        0       0        0
iostat_ex_8.d2          0      0        0      0        0       0        0
iostat_ex_9.d1          0      0        0      0        0       0        0
iostat_ex_9.d2          0      0        0      0        0       0        0
iostat_ex_10.d1         0      0        0      0        0       0        0
iostat_ex_10.d2         0      0        0      0        0       0        0
iostat_ex_11.d1         0      0        0      0        0       0        0
iostat_ex_11.d2         0      0        0      0        0       0        0
iostat_ex_12.d1         0      0        0      0        0       0        0
iostat_ex_12.d2         0      0        0      0        0       0        0
iostat_ex.a1            0      0        0      0        0       0        0
iostat_ex.a2            0      0        0      0        0       0        0
iostat_ex.a3            0      0        0      0        0       0        0
```

26

PROSTRCT Utility

This chapter describes the OpenEdge database administration utility PROSTRCT.

For details, see the following topics:

- PROSTRCT utility
- PROSTRCT ADD qualifier
- PROSTRCT ADDONLINE qualifier
- PROSTRCT BUILDDB qualifier
- PROSTRCT CLUSTER CLEAR qualifier
- PROSTRCT CREATE qualifier
- PROSTRCT LIST qualifier
- PROSTRCT REMOVE qualifier
- PROSTRCT REORDER AI qualifier
- PROSTRCT REPAIR qualifier
- PROSTRCT STATISTICS qualifier
- PROSTRCT UNLOCK qualifier

PROSTRCT utility

Creates and maintains an OpenEdge database. For example, you can use PROSTRCT and its qualifiers to perform the following tasks:

- Create a new database from an existing structure description (`.st`) file
- Display storage usage statistics including information about storage areas in a database
- Add areas and extents to a database
- Remove areas and extents from a database

Syntax

```
prostrct qualifier db-name [ structure-description-file ]
     [ [-userid username [-password passwd ] ] |
      [ -U username -P passwd]
     ]
     [-Passphrase]
```

Parameters

qualifier

Specifies the qualifier that you want to use. You can supply the following qualifiers:

qualifier

Specifies the qualifier that you want to use. You can supply the following qualifiers:

- ADD
- ADDONLINE
- BUILDDB
- CLUSTER CLEAR
- CREATE
- LIST
- REMOVE
- REORDER AI
- REPAIR
- STATISTICS
- UNLOCK

Details of the qualifiers are found in the PROSTRCT subsections listed in the following pages.

db-name

Specifies the database you are using.

structure-description-file

Specifies the new `.st` file. Do not use `dbname.st`.

`-userid` *username* [`-password` *passwd*] | `-U` *username* `-P` *passwd*

A pair of username and password parameters are used together to support a common user authentication model to specify the user ID and password when connecting to an OpenEdge RDBMS.OpenEdge authenticates the user ID and password against the user accounts defined by the authentication system configured for the user's security domain.

Use username and password parameters to identify an authenticated database administrator.

If you do not specify a user ID and password for a database connection, OpenEdge assigns the OS process's user-id as the default user-id.

For more information on authentication systems and security domains, see *OpenEdge Getting Started: Identity Management*.

`-Passphrase`

For encryption-enabled databases only, specifies to prompt for a passphrase to authenticate the key store. Utilities that open the database must at a minimum authenticate as the key store user. See Transparent Data Encryption on page 199 for more information on encryption and key store authentication.

Notes

- You must define your database structure in a `.st` file before you create a database. PROSTRCT CREATE creates a database control (DB) area from the information in the `.st` file.
- See Creating and Deleting Databases on page 41 for a complete description of structure description files and storage areas.
- PROSTRCT supports the use of internationalization startup parameters such as `-cpinternal`*codepage* and `-cpstream`*codepage*. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.

PROSTRCT ADD qualifier

Appends the files from a new structure description (`.st`) file to a database.

Syntax

```
prostrct add db-name [ structure-description-file ] [ -validate ]
```

Parameters

db-name

Specifies the database you are using.

structure-description-file

Specifies the new `.st` file. Do not use *db-name*`.st`.

`-validate`

Parses the contents of the `.st` file for syntax errors. When `-validate` is specified, the database is not created. Each line of the structure file is read and evaluated. If errors are detected, the type of error is reported along with the line number. For a discussion of the syntax and rule checking `-validate` provides, see Validating structure description files on page 341.

Use PROSTRCT ADD to add new storage areas and extents to new or existing storage areas.

Notes

- You can use PROSTRCT ADD to add areas and extents only when the database is offline. Use PROSTRCT ADDONLINE to add areas and extents to an online database. For details, see PROSTRCT ADDONLINE qualifier on page 788.
- The new structure description file cannot identify existing extent files. It can only contain the definitions for new extent files. See Creating and Deleting Databases on page 41 for a complete description of structure description files and storage areas.
- Run PROSTRCT LIST after updating the database with PROSTRCT ADD. PROSTRCT LIST verifies that the `.st` contains the updated information.

PROSTRCT ADDONLINE qualifier

Appends the files from a new structure description (`.st`) file to an online database.

Syntax

```
prostrct addonline db-name [ structure-description-file ] [ -validate ]
```

Parameters

db-name

Specifies the database you are using.

structure-description-file

Specifies the new `.st` file. Do not use *db-name*`.st`.

`-validate`

Parses the contents of the `.st` file for syntax errors. When `-validate` is specified, the database is not created. Each line of the structure file is read and evaluated. If errors are detected, the type of error is reported along with the line number. For a discussion of the syntax and rule checking `-validate` provides, see Validating structure description files on page 341.

Use PROSTRCT ADDONLINE to add new storage areas and extents to new or existing storage areas to an online database.

Notes

- PROSTRCT ADDONLINE has the following restrictions:
  - You may not have more than one instance of PROSTRCT ADDONLINE executing at a single time.
  - If currently connected users will not have sufficient privileges to access the new extents, you may proceed with the online add, but an under-privileged user will be disconnected from the database the first time it tries to open one of the new extents. This open may occur at an unpredictable time, such as when the user needs to find space in the buffer pool. PROSTRCT ADDONLINE verifies the status of current users before adding new extents, and prompts you to confirm that you wish to proceed.
- If your database has after-imaging enabled, you must update your target database to match the source database. Failure to update the target database will cause RFUTIL ROLL FORWARD to fail when applying after-image extents. PROSTRCT ADDONLINE issues a warning, and prompts you to continue as follows:

```
Adding a new area online with ai enabled will cause the roll forward utility to fail
 unless the corresponding areas are also added to the target database. (13705)
Do you want to continue (y/n)? (13706)y
```

- If ADDONLINE fails, the newly created files will be deleted.
- For a discussion and examples of PROSTRCT ADDONLINE, see Using the OpenEdge Structure Add utility on page 329.

PROSTRCT BUILDDB qualifier

Re-creates a control area from the structure description (`.st`) of an existing database.

Syntax

```
prostrct builddb db-name [ structure-description-file ]
```

Parameters

db-name

Specifies the database you are using.

structure-description-file

Specifies the existing `.st` file.

Use to recover when an existing database control (`.db`) area is lost or damaged.

Caution: It is important to maintain an up to date structure definition file. Whenever you add extents to your database with PROSTRCT ADD, be sure to update your structure file with PROSTRCT LIST. If your structure definition file is not accurate, the database control area built with PROSTRCT BUILDDB will not be accurate.

Note

BUILDDB does only minimal validation of the resulting control area.

PROSTRCT CLUSTER CLEAR qualifier

Clears the cluster setting in the master block of the database. Clearing the cluster setting allows you to start and manage the database without cluster protection.

Syntax

```
prostrct cluster db-name clear
```

Parameters

db-name

Specifies the database you are using.

Notes

- Use of the PROSTRCT command to clear the cluster setting is for emergency use only. Under normal circumstances, the PROCLUSTER command should be used to disable a cluster-enabled database. See Disabling a cluster-enabled database on page 244.
- Clearing the cluster setting does not clean up cluster-specific objects associated with the database. You must manually remove these objects.

PROSTRCT CREATE qualifier

Creates a void OpenEdge database from a previously defined structure description (`.st`) file.

Syntax

```
prostrct create db-name  [ structure-description-file ]
                         [ -blocksize blocksize ]
                         [ -validate ]
```

Parameters

db-name

Specifies the database you want to create.

structure-description-file

Specifies the `.st` file you want PROSTRCT to access for file information. The default `.st` file is `db-name.st`. If you have a structure description file that has the same name as the database you

are creating, you do not have to specify *structure-description-file*. PROSTRCT automatically creates the database from the structure description file that has the same name as your database with the extension `.st`.

`- blocksize` *blocksize*

Specifies the database block size in bytes (for example `-blocksize 1024`). The maximum number of indexes allowed in a database is based on the database block size. See OpenEdge RDBMS Limits on page 59 for more information on database limits.

`-validate`

Parses the contents of the `.st` file for syntax errors. When `-validate` is specified, the database is not created. Each line of the structure file is read and evaluated. If errors are detected, the type of error is reported along with the line number. For a discussion of the allowed syntax in a description file, see Creating a structure description file on page 43

The PROSTRCT CREATE utility allows you to specify the minimum amount of information necessary to create a database. You must specify the area type and extent location. If the extent is fixed length, you must also specify the size. You need not provide specific filename or file extensions. The utility will generate filename and file extensions for all database files according to the following naming convention:

- The control area (`.db`) and the log file (`.lg`) are placed in the directory specified by the command line *db-name* parameter.
- If a relative pathname is provided, including using common "." (dot) notation, the relative pathname will be expanded to an absolute pathname. Relative paths begin in your current working directory.
- For BI extents, the filename is the database name with a .b*n* extension, where *n* represents the number of extents created.
- For AI extents, the filename is the database name with a .a*n* extension, where *n* represents the logical order in which the AI areas will be accessed during normal processing.
- For TL extents, the filename is the database name, with a .t*n* extension.
- For Schema area extents, the filename is the database name with a .d*n* extension, where *n* represents the order extents were created and will be used.
- For application data extents, the filename is the database name and an area number. The area number is a unique identifier that differentiates between different areas. User extent filenames also have a .d*n* extension, where *n* represents the order extents were created and will be used.

The newly created database does not contain any metaschema information. Rather, it consists of the database control (DB) area and whatever primary recovery (BI), after-image (AI), two-phase commit transaction log (TL), and application data (`.dn`) areas you defined in the `.st` file.

After you create a void database, you must add metaschema information. The OpenEdge RDBMS provides empty databases, each the size of a supported database block size. The empty database and the database you want to copy it to must have the same block size.

Caution: Never use operating system file commands to copy an OpenEdge database. Instead, use the PROCOPY or PRODB utilities.

Notes

- You cannot create a database if one already exists with the same name.
- See Maintaining Database Structure on page 325 for more information about the CREATE qualifier with the PROSTRCT utility.

PROSTRCT LIST qualifier

Creates a structure description (`.st`) file for an OpenEdge database.

Syntax

```
prostrct list db-name [ structure-description-file ]
```

Parameters

db-name

Specifies the multi-volume database whose structure description file you want to update.

structure-description-file

Specifies the structure description file PROSTRCT creates. If you do not specify the structure description file, PROSTRCT uses the base name of the database and appends a `.st` extension. It replaces an existing file of the same name.

PROSTRCT LIST provides storage area, transaction log, and records per block information in the `.st` file it produces. Also, PROSTRCT LIST displays storage area names and extent information including the extent type, size, number, and name.

Notes

- Use PROSTRCT LIST any time you make changes to the structure of the database to verify that the change was successful.
- You can use this utility with an online database.
- See Maintaining Database Structure on page 325 for more information about the LIST qualifier with the PROSTRCT utility.

PROSTRCT REMOVE qualifier

Removes storage areas or extents within storage areas.

Syntax

```
prostrct remove db-name extent-token storage-area
```

Parameters

db-name

Specifies the database where you want to remove a storage area or extent.

extent-token

Indicates the type of extent to remove. Specify one of the following:

- `d` — Removes a data extent
- `bi` — Removes a before-image extent
- `ai` — Removes an after-image extent
- `tl` — Removes a transaction log extent

storage-area

Specifies the name of the storage area to remove.

Notes

- You cannot remove an extent if it is in use or contains any data.
- You must truncate the bi file before removing an extent.
- Before you remove extents from the TL area, you must disable two-phase commit.
- You must disable after-imaging to remove variable, or fixed-length AI extents.
- You can verify that the deletion occurred and update the structure description file using the LIST qualifier with the PROSTRCT utility.
- See Maintaining Database Structure on page 325 for more information about the REMOVE qualifier with the PROSTRCT utility.

PROSTRCT REORDER AI qualifier

Reorders AI extents.

Syntax

```
prostrct reorder ai db-name
```

Parameters

db-name

Specifies the database where you want to reorder AI extents.

Notes

- Reorder AI extents only if you cannot empty extents, perhaps because of OpenEdge Replication locking an extent.
- REORDER AI will move empty AI extents to follow the busy extent in switch order.
- REORDER AI will rename AI extents and associate them with different extent areas. Use PROSTRCT LIST to generate a new `.st` file for your database immediately after reordering. For more information on generating a new `.st` file, see PROSTRCT LIST qualifier on page 792.
- See Managing after-imaging files on page 144 for more information about the REORDER AI qualifier.

PROSTRCT REPAIR qualifier

Updates a database's control information after an extent is moved or renamed.

Syntax

```
prostrct repair db-name [ structure-description-file ]
```

Parameters

db-name

Specifies the database you are using.

structure-description-file

Specifies the `.st` file containing the updated extent information. If you omit the *structure-description-file*, PROSTRCT REPAIR uses the *db-name*`.st` file to update the control information.

Notes

- Start with a current copy of the database `.st` file.
- The `.st` extension on the new structure description file is optional.
- You cannot use the REPAIR qualifier to add or remove extents. You can only use it to change the location of existing extents.
- You must manually move the `.db` file or the data extent. PROSTRCT REPAIR simply updates the file list of the `.db` file to reflect the new locations of database extents.

PROSTRCT STATISTICS qualifier

Displays statistics about storage information about the specified database.

Syntax

```
prostrct statistics db-name
```

Parameters

db-name

Specifies the database whose storage information you want.

PROSTRCT STATISTICS displays the following information:

- Database name

- Primary database block size and the before-image and after-image block sizes
- Storage area and each extent within it
- Total number of active blocks allocated for each data area
- Total number of blocks for each data area
- Total number of empty blocks in each data area
- Total number of extent blocks for each data area
- Total records per block for each data area
- Total number of all blocks (active blocks, active free blocks, and empty blocks)
- Date and time of the last full database backup

Notes

- When PROSTRCT STATISTICS is run against a database that is in use, values may be changing as the display is output.
- Do not run this utility against a database that has crashed. You must recover the database first.
- A void database cannot be accessed by this command.
- See Maintaining Database Structure on page 325 for more information about the STATISTICS qualifier with the PROSTRCT utility.

PROSTRCT UNLOCK qualifier

Allows you to access a damaged database structure and correct inconsistencies between the creation date and the open date in the database file header block and the master block. This is especially useful when trying to force open a database or when recovering a lost database file.

Caution: Use this utility as a last resort. Call Technical Support for further direction.

Syntax

```
prostrct unlock db-name  [ -extents ]
```

Parameters

db-name

Specifies the database where you want to force access.

-extents

Replaces missing extents with empty extents if any database files are missing.

Notes

- When the database finds an inconsistency among the data and recovery log, it generates an error message and stops any attempt to open the database. Typically, inconsistencies between files are a result of accidental

misuse of operating system copy utilities, deletion mistakes, or incorrectly administered backup and restore procedures.

- If the first data file (`.d1`) is missing, the database cannot open because of the missing master block. PROSTRCT UNLOCK with the `-extents` parameter, however, creates an empty file with the same name and location as the missing file that allows the database to open. This function helps enable access to a severely damaged database.
- PROSTRCT UNLOCK does not repair damaged databases. It opens databases with inconsistencies in dates and missing extents, but these databases still need to be repaired before they can be used. For information on repairing databases, see Restoring a database on page 111.

27

RFUTIL Utility

This chapter describes the OpenEdge database administration utility RFUTIL.

For details, see the following topics:

- RFUTIL utility syntax
- RFUTIL AIARCHIVER DISABLE qualifier
- RFUTIL AIARCHIVER ENABLE qualifier
- RFUTIL AIARCHIVER END qualifier
- RFUTIL AIARCHIVER SETDIR qualifier
- RFUTIL AIARCHIVER SETINTERVAL qualifier
- RFUTIL AIARCHIVE EXTENT qualifier
- RFUTIL AIARCHIVE NEXTEXTENT qualifier
- RFUTIL AIMAGE AIOFF qualifier
- RFUTIL AIMAGE BEGIN qualifier
- RFUTIL AIMAGE END qualifier
- RFUTIL AIMAGE EXTENT EMPTY qualifier
- RFUTIL AIMAGE EXTENT FULL qualifier
- RFUTIL AIMAGE EXTENT LIST qualifier
- RFUTIL AIMAGE EXTRACT qualifier

- RFUTIL AIMAGE NEW qualifier
- RFUTIL AIMAGE QUERY qualifier
- RFUTIL AIMAGE SCAN qualifier
- RFUTIL AIMAGE TRUNCATE qualifier
- RFUTIL AIVERIFY PARTIAL qualifier
- RFUTIL AIVERIFY FULL qualifier
- RFUTIL MARK BACKEDUP qualifier
- RFUTIL ROLL FORWARD qualifier
- RFUTIL ROLL FORWARD OPLOCK qualifier
- RFUTIL ROLL FORWARD RETRY qualifier
- RFUTIL ROLL OPUNLOCK qualifier
- RFUTIL SEQUENCE qualifier

RFUTIL utility syntax

Performs various roll-forward recovery activities, depending on the qualifier you supply.

Syntax

```
rfutil db-name -C qualifier
     [ [-userid username [-password password] ] |
       [ -U username -P password ] ]
         [-Passphrase]
```

Parameters

db-name

Specifies the database you are using.

-C

Specifies a particular utility or function when you use the RFUTIL command.

qualifier

Specifies the qualifier that you want to use. You can supply the qualifiers listed in Table 108: RFUTIL utility qualifiers on page 799.

`-userid` *username* [`-password` *passwd*] `-U` *username* `-P` *passwd*

A pair of username and password parameters are used together to support a common user authentication model to specify the user ID and password when connecting to an OpenEdge RDBMS.OpenEdge authenticates the user ID and password against the user accounts defined by the authentication system configured for the user's security domain.

Use username and password parameters to identify an authenticated database administrator.

If you do not specify a user ID and password for a database connection, OpenEdge assigns the OS process's user-id as the default user-id.

For more information on authentication systems and security domains, see *OpenEdge Getting Started: Identity Management*.

`-Passphrase`

For encryption-enabled databases only, directs RFUTIL to prompt for a passphrase to authenticate the key store. Utilities that open the database must at a minimum authenticate as the key store user. Utilities that modify encryption policies (such as enabling or disabling encryption of after-image files) must authenticate as the key store admin. See Transparent Data Encryption on page 199 for more information on transparent data encryption and key store authentication.

If your database is enabled for transparent data encryption and configured for manual start, you must specify `-Passphrase` every time the database is opened.

Note: The `-passphrase` parameter will be required for every utility that modifies encryption policies on databases enabled with transparent data encryption.

Note: RFUTIL and its qualifiers support the use of internationalization startup parameters such as `-cpinternal codepage` and `-cpstream codepage`. See Database Startup Parameters on page 473 for a description of database-related internationalization startup parameters.

The table below lists the RFUTIL qualifiers. See the appropriate section for each of the qualifiers for more detailed information.

Table 108: RFUTIL utility qualifiers

Qualifier	Description
AIARCHIVER DISABLE	Disables the AI File Management Utility.
AIARCHIVER ENABLE	Enables the AI File Management Utility.
AIARCHIVER END	Shuts down the AI File Management Utility daemon.
AIARCHIVER SETDIR	Changes the archive destination directory list for the AI File Management Utility daemon.
AIARCHIVER SETINTERVAL	Changes the value of the archive interval for the AI File Management daemon.
AIARCHIVE EXTENT	Manually archives the specified AI extent when AI File Management is not running.

Qualifier	Description
AIMAGE AIOFF	Disables after-imaging for a database during maintenance.
AIMAGE BEGIN	Enables after-imaging for a database.
AIMAGE END	Disables after-imaging for a database.
AIMAGE EXTENT EMPTY	Marks an AI extent as empty and informs the database manager that the indicated AI extent has been manually backed up and is now free for reuse.
AIMAGE EXTENT FULL	Displays the pathname of the oldest filled file.
AIMAGE EXTENT LIST	Displays information about extents.
AIMAGE EXTRACT	Extracts the active blocks of an AI extent and writes them to a file.
AIMAGE NEW	Switches after-imaging to the next AI extent.
AIMAGE QUERY	Queries an AI extent for status.
AIMAGE SCAN	Scans AI files and displays information for the files.
AIMAGE TRUNCATE	Truncates all of the variable-length AIextents and optionally sets the AI block size with the After-image Block Size (`-aiblocksize`) parameter.
AIVERIFY PARTIAL	Verifies the AI extent by validating AI files before they are applied. This verification focuses on the AI block and the `rlnote` structure (note headers), while providing additional information in error messages that specify issues found in the AI extent.
AIVERIFY FULL	Verifies the full AI extent, including verification of note data members.
MARK BACKEDUP	Marks the database file, indicating that you have just completed a backup of the database.
ROLL FORWARD	Reconstructs a database by applying to that database all notes stored in the AI file.
ROLL FORWARD OPLOCK	Enforces a protection mechanism during the roll forward process, so that there will be no interruption to the target database before the final roll forward session is completed.
ROLL FORWARD RETRY	Restarts the roll-forward operation on the after-image extent that was in the process of rolling forward. The retry operation finds the transaction in process at the time of failure and resumes rolling forward.
ROLL OPUNLOCK	Cancels the roll forward protection established by ROLL FORWARD OPLOCK.
SEQUENCE	Updates the after-image sequence number of a database.

Note: You cannot use RFUTIL against a database in single-user mode.

The RFUTIL entries that follow describe each of these qualifiers.

RFUTIL AIARCHIVER DISABLE qualifier

Disables AI File Management for the automated management of AI files. Use this qualifier only when after-imaging is enabled, and AI File Management is enabled.

Syntax

```
rfutil db-name -C aiarchiver disable
```

Parameters

db-name

Specifies the database you are using.

Notes

- After-imaging is not disabled with this command, only the AI File Management Utility is disabled.
- If the database is online and the AI File Management daemon is running, the daemon is first shut down, then AI File Management is disabled.

RFUTIL AIARCHIVER ENABLE qualifier

Activates AI File Management for the automated management of AI files. Use this qualifier only when after-imaging is enabled.

Syntax

```
rfutil db-name -C aiarchiver enable
```

Parameters

db-name

Specifies the database you are using.

Notes

- Issuing AIARCHIVER ENABLE has the following impact:
  - The database broker automatically starts the AI File Management daemon when the broker starts.
  - The AI Archive log file (*database-name*`.archival.log`) is created in the same directory as your database (`.db`) file.

- Your database must be offline when this command is issued.
- Earlier versions of OpenEdge are not allowed to directly connect to your database when AI File Management is enabled. Clients may still connect using a network connection.

Operation of the AI File Management daemon is controlled through other AIARCHIVER qualifiers to RFUTIL, PROSERVE, and PROBKUP. See AI File Management utility on page 150 for details.

RFUTIL AIARCHIVER END qualifier

Stops the AI File Management Utility daemon process, if it is running.

Syntax

```
rfutil db-name -C aiarchiver end
```

Parameters

db-name

Specifies the database you are using.

Notes

- Use this command to stop the AI File Management Utility daemon without shutting down the database server. This command does not stop after-image processing. The AI extents continue to fill as part of normal database activity.
- See After-imaging on page 139for instructions on restarting the daemon.

RFUTIL AIARCHIVER SETDIR qualifier

Changes the directory list used by the AI File Management Utility daemon process.

Syntax

```
rfutil db-name -C aiarchiver setdir dir-list [ -aiarcdircreate ]
```

Parameters

db-name

Specifies the database you are using.

dir-list

Specifies a comma-separated list of directories used as target locations for archived AI files. The directories must exist, unless you specify `-aiarcdircreate`. The specified *dir-list* replaces

any previously specified directory list, and the directory names may not include embedded spaces or commas. The total length of *dir-list* cannot exceed 512 characters.

`-aiarcdircreate`

Directs the daemon to create the directories specified by the `-aiarcdir` parameter, if they do not currently exist.

Notes

Use this command against a database that is online with an active AI File Management Utility daemon.

RFUTIL AIARCHIVER SETINTERVAL qualifier

Changes the archive interval used by the AI File Management Utility daemon process.

Syntax

```
rfutil db-name -C aiarchiver setinterval n
```

Parameters

db-name

Specifies the database you are using.

n

Specifies the operation mode of the daemon process and the time the daemon sleeps between mandatory extent switches in timed mode.

When *n* is a value between 120 and 86,400, the daemon executes in timed mode, and *n* indicates the length of time, in seconds, that the daemon waits between mandatory extent switches.

Notes

Use this command against a database that is online with an active AI File Management Utility daemon.

RFUTIL AIARCHIVE EXTENT qualifier

Manually archives the specified AI extent.

Syntax

```
rfutil db-name -C aiarchive extent source-extent target
```

Parameters

db-name

Specifies the database you are using.

source-extent

Specifies the AI extent to archive. Supply the full file name.

target

Specifies the output file of the archived extent. The file cannot exist. If a file exists with the specified name, AIARCHIVE EXTENT fails.

Notes

- This command works against both online and offline databases.
- This command does not require AI File Management to be enabled.
- If the database is online and AI File Management is enabled, the daemon process cannot be running when this command is executed.
- If AI File Management is enabled, the extent archive is logged to the AI File Management log file (*database-name*.archival.log.) If AI File Management is not enabled, the extent is still archived, but the execution is not logged.
- This command implicitly empties the AI extent that is archived, marking it EMPTY.

RFUTIL AIARCHIVE NEXTEXTENT qualifier

Manually archives the next FULL AI extent.

Syntax

```
rfutil db-name -C aiarchive nextextent
```

Parameters

db-name

Specifies the database you are using.

AIARCHIVE NEXTEXTENT forces the next FULL AI extent to be archived immediately. The extent is archived to the previously specified directory list. If the daemon is operating in timed mode, the timer is reset. If your database stalls because all your AI extents are full, this command will free an extent. After clearing the stall, set the archive interval to a smaller number or to 0 prevent another stall.

Notes

- This command works against both online and offline databases.
- This command does not require AI File Management to be enabled.

- If the database is online and AI File Management is enabled, the daemon process cannot be running when this command is executed.
- If AI File Management is enabled, the extent archive is logged to the AI File Management log file (*database-name*.archival.log.) If AI File Management is not enabled, the extent is still archived, but the execution is not logged.
- This command implicitly empties the AI extent that is archived, marking it EMPTY.

RFUTIL AIMAGE AIOFF qualifier

Disables after-imaging for a database in multi-user mode.

Syntax

```
rfutil db-name -C aimage aioff [-aiencryption disable]
```

Parameters

db-name

Specifies the database you are using.

-aiencryption disable

For databases enabled for Transparent Data Encryption only, -aiencryption disable reiterates the directive to disable encryption of your AI files. You cannot specify -aiencryption disable for a Replication target database.

You do not have to shut down the database to use AIMAGE AIOFF. Use AIMAGE AIOFF to:

- Perform scheduled maintenance on a database. Disabling after-imaging during a table move or index rebuild saves time and disk space.
- Prevent database crashes caused by a pending lack of disk space. Instead of switching AI extents, you can disable after-imaging.

Caution: If you disable after-imaging and the database crashes, you cannot roll forward.

Notes

- AIMAGE OFF requires an Enterprise Database License.
- AIMAGE OFF cannot be performed if the database is an OpenEdge Replication source database and the replication server is running.
- Disabling after-imaging with AIMAGE OFF disables AI Management and Replication if either is active.
- Disabling after-imaging with AIMAGE OFF disables AI encryption, unless your database is a Replication target database.
- If the database is online, re-enable after-imaging with PROBKUP. See PROBKUP utility on page 832 for details.
- If the database is offline, re-enable after-imaging with RFUTIL AIMAGE BEGIN.

RFUTIL AIMAGE BEGIN qualifier

Enables after-imaging for a database.

Syntax

```
rfutil db-name -C aimage begin [ -aiencryption enable | disable ]
```

Parameters

db-name

Specifies the database you are using.

`-aiencryption enable | disable`

For databases with Transparent Data Encryption enabled only, specifies whether to enable or disable encryption of your AI files. If not specified, your AI files **will be encrypted**. Progress Software Corporation strongly recommends you encrypt your AI files.

The AIMAGE BEGIN qualifier:

- Creates the AI area that holds after-image notes
- Labels the database as having after-imaging enabled

The AIMAGE BEGIN qualifier fails if:

- It cannot truncate an existing AI area.
- You have not marked the database as backed up since the last time the database was modified.

RFUTIL AIMAGE END qualifier

Disables after-imaging for a database.

Syntax

```
rfutil db-name -C aimage end [-aiencryption disable ]
```

Parameters

db-name

Specifies the database you are using

`-aiencryption disable`

For databases enabled for Transparent Data Encryption only, `-aiencryption disable` reiterates the directive to disable encryption of your AI files.

Note: You cannot specify `-aiencryption disable` for a Replication target database.

Notes

- The AIMAGE END qualifier fails if after-imaging is not enabled.
- Disabling after-imaging with AIMAGE END disables AI Management and Replication if either is active.
- Disabling after-imaging with AIMAGE END disables AI encryption, unless your database is a Replication target database.
- For information on re-enabling after-imaging:
  - See PROBKUP utility on page 832, for enabling after-imaging when the database is online.
  - See RFUTIL AIMAGE BEGIN qualifier on page 806, for enabling after-imaging when the database is offline.

RFUTIL AIMAGE EXTENT EMPTY qualifier

Marks an AI extent as empty and informs the database manager that the indicated AI extent has been manually backed up and is now free for reuse.

Syntax

```
rfutil db-name -C aimage extent empty [ extent-number | extent-path ]
```

Parameters

db-name

Specifies the database you are using.

extent-number

Specifies the number of the extent you want to mark as empty.

extent-path

Specifies the pathname of the extent you want to mark as empty.

Notes

- Use RFUTIL AIMAGE EXTENT LIST or RFUTIL AIMAGE EXTENT FULL to determine the *extent-number* or *extent-pathname*.
- If you do not specify either an extent number or an extent path, RFUTIL marks the oldest full extent as empty.

- If the extent being marked empty is a variable length extent, RFUTIL will truncate the extent.
- If an extent is marked LOCKED by OpenEdge Replication, it cannot be emptied with this command.
- This command cannot be run against a database that has enabled AI File Management.

RFUTIL AIMAGE EXTENT FULL qualifier

Displays the pathname of the oldest filled file; this is the next file to be backed up.

Syntax

```
rfutil db-name -C aimage extent full
```

Parameters

db-name

Specifies the database you are using.

Notes

Use RFUTIL AIMAGE EXTENT FULL in a script similar to the following:

```
last_full='_rfutil mydb -C aimage extent full'
tar -cvf /dev/ai_archive $last_full
rfutil mydb -C aimage extent full $last_full
```

RFUTIL AIMAGE EXTENT LIST qualifier

Displays the status of the database AI extents.

Syntax

```
rfutil db-name -C aimage extent list
```

Parameters

db-name

Specifies the database you are using.

AIMAGE EXTENT LIST displays the following information:

- **Extent Number** — Number of each file
- **Extent Type** — Type of each file — either fixed length or variable length
- **Extent Path** — Pathname of each file

- **Extent Size** — Size of each file in 1K blocks
- **Space Used** — Number of blocks of space used in each file
- **Extent Status** — Status of each file — either empty, full, or busy.
- **Start/Date Time** — Time each file began logging AI notes — not applicable to empty files

Notes

The status of a file might be full even though there is space left over in the file. This can happen after an online backup because file switch-over occurs at the time of the backup, whether or not the current file is full. RFUTIL still marks the file as full because, like a full file, it must be archived and marked empty before the database can reuse it.

RFUTIL AIMAGE EXTRACT qualifier

Extracts the active blocks of the specified AI extent and writes them to an output file.

Syntax

```
rfutil db-name -C aimage extract -a ai-extent -o output-file
```

Parameters

db-name

Specifies the database you are using.

-a *ai-extent*

Identifies the AI file containing the blocks to be extracted. The extent must be marked as full or locked.

-o *output-file*

Specifies the file to hold the extracted blocks.

Notes

- Extracting blocks from an AI extent is only beneficial for fixed length extents. There will be minimal savings of disk space when extracting blocks from a variable length extent.
- See Managing after-imaging files on page 144 for more information on extracting AI blocks from an extent.

RFUTIL AIMAGE NEW qualifier

Changes the status of the current AI extent to full and changes the status of the next AI extent to busy.

Syntax

```
rfutil db-name -C aimage new
```

Parameters

db-name

Specifies the database you are using.

Notes

- The AIMAGE NEW qualifier fails if:
  - The next extent is not empty
  - The database has no AI extents
- You can use this qualifier whether the database is offline or online.
- Use this qualifier only when after-imaging is enabled and you have just backed up the AI area.

RFUTIL AIMAGE QUERY qualifier

Queries an AI file and displays information from the query.

Syntax

```
rfutil db-name -C aimage query query-option by search-optionsearch-value
```

Parameters

db-name

Specifies the database you are using.

query-option

Specifies the information you want to gather about the AI extent. You can supply one of the AIMAGE QUERY options from the table that follows.

Table 109: AIMAGE QUERY options

Query option	Value returned
EXTNUM	The extent number
STATUS	The status of the extent: FULL, LOCKED, BUSY, EMPTY
TYPE	The type of file: Fixed Length or Variable Length

Query option	Value returned
SIZE	The size of the extent in 1K blocks
USED	The amount of the extent used in 1K blocks
NAME	The full filename of the extent
SEQUENCE	The sequence number
STARTDATE	The date and time after-imaging started writing to the extent
ALL	All of the above information

search-option

Specifies how you are identifying the AI extent to query. Supply one of the options from column one of Table 110: AIMAGE QUERY search options and values on page 811.

search-value

Specifies the match criteria for the *search-option*. Supply the value indicated in column two of Table 110: AIMAGE QUERY search options and values on page 811 that matches your *search-option*.

Table 110: AIMAGE QUERY search options and values

Search-option	Search-value
EXTNUM	Integer value, in a file named, dbname.a*n*, the EXTNUM is the value *n*
NAME	Character string, the extent file name
SEQUENCE	Integer value, the sequence number of the extent

Notes

- The AIMAGE QUERY qualifier returns minimal output.
- The output of the AIMAGE QUERY qualifier is intended to be used in automated after-imaging procedures.

RFUTIL AIMAGE SCAN qualifier

Scans AI files and displays information from those files.

Syntax

```
rfutil db-name -C aimage scan [ verbose ] {-a ai-name | -ailist listfile}
```

Parameters

db-name

Specifies the database you are using.

verbose

Provides more information from the AI area, including the transaction number, the date and time the transaction began or ended, and the user ID of the user who initiated the transaction. You might want to try this on a small test AI area before running it on the after-image file associated with your database.

-a *ai-name*

Identifies the AI area of the specified database.

-ailist *listfile*

Identifies a file containing a list of AI area files. The *listfile* file must be structured as follows:

- Specify one AI file per line
- AI file names can be fully specified absolute file name or a relative file name relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

Note: If you specify a list of AI files with -ailist and a file is **not** an after-image file, a warning is issued and the file is skipped.

Notes

- The AIMAGE SCAN qualifier fails if:
  - You omit the After-image Filename (-a) or AI List (-ailist) parameter
  - It cannot open the AI area
- The specified database does not have to be the database that corresponds to the AI area. You can use a dummy database to use this command with an AI area for an online database.

RFUTIL AIMAGE TRUNCATE qualifier

Truncates all of the variable-length AIextents and optionally sets the AI block size with the After-image Block Size (-aiblocksize) parameter.

Syntax

```
rfutil db-name -C aimage truncate -aiblocksize size
```

Parameters

db-name

Specifies the database you are using.

-aiblocksize *size*

Specifies the size of the AI blocks in each buffer, in kilobytes. The valid values are 1, 2, 4, 8, and 16. The block size cannot be smaller than the database block size.

Notes

- After running this command to change the AI block size, you must perform a full backup of the database before you can re-enable after-imaging. If you change the BI block size or cluster size before backing up the database, the block size of the backup will overwrite the changed block size when the backup is restored.
- Increasing the AI block size allows larger AI reads and writes. This can reduce I/O rates on disks where the AI areas are located. If your operating system can benefit from larger writes, this option can improve performance. Larger AI block size might also improve performance for roll-forward recovery processing.
- When you run this command, after-imaging and two-phase commit must be disabled, and the database must be offline; otherwise, RFUTIL returns an error message.
- After you change the AI block size, RFUTIL uses the new block size in all database operations.
- Use the PROSTRCT STATISTICS qualifier to display the block sizes for a database.
- Typically, if you change the AI block size, you should also change the before-image (BI) block and cluster size; otherwise, the increased AI performance will cause a BI bottleneck.
- See Managing Performance on page 275 for more information about using the RFUTIL AIMAGE TRUNCATE utility.

RFUTIL AIVERIFY PARTIAL qualifier

Verifies the AI extent by validating AI files before they are applied. This verification focuses on the AI block and the `rlnote` structure (note headers), while providing additional information in error messages that specify issues found in the AI extent.

Syntax

```
rfutil db-name -C aiverify partial {-a ai-name | -ailist listfile}
```

Parameters

db-name

Specifies the database you are using.

-a *ai-name*

Identifies the AI extent name.

`-ailist` *listfile*

Identifies a file containing a list of AI area files. The *listfile* file must be structured as follows:

- Specify one AI file per line
- AI file names can be fully specified absolute file name or a relative file name relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

Note: If you specify a list of AI files with `-ailist` and a file is **not** an after-image file, a warning is issued and the file is skipped.

RFUTIL AIVERIFY FULL qualifier

Verifies the full AI extent by validating AI files before they are applied. In addition to focusing on the AI block and the `rlnote` structure (note headers), this qualifier verifies note data members.

Syntax

```
rfutil db-name -C aiverify full { -a ai-name | -ailist listfile }
```

Parameters

db-name

Specifies the database you are using.

`-a` *ai-name*

Identifies the AI extent name.

`-ailist` *listfile*

Identifies a file containing a list of AI area files. The *listfile* file must be structured as follows:

- Specify one AI file per line
- AI file names can be fully specified absolute file name or a relative file name relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

Note: If you specify a list of AI files with `-ailist` and a file is **not** an after-image file, a warning is issued and the file is skipped.

RFUTIL MARK BACKEDUP qualifier

Marks the database file, indicating that you have just completed a backup of the database.

Syntax

```
rfutil db-name -C mark backedup
```

Parameters

db-name

Specifies the database you want to mark as backed up.

Use the MARK BACKEDUP qualifier if you are using an operating system backup utility instead of PROBKUP. The Backup and Restore utilities automatically mark the database as backed up.

RFUTIL ROLL FORWARD qualifier

Reconstructs a database by applying to that database all notes stored in the AI file.

Syntax

```
rfutil db-name -C roll forward [ verbose ]
      [    endtime yyyy:mm:dd:hh:mm:ss | endtrans transaction-number]
      [ -B n] [ -r ] {-a ai-name | -ailist listfile}
```

Parameters

db-name

Specifies the database you are using.

verbose

Produces information for every note in the AI area.

-a *ai-name*

Identifies the AI area of the specified database.

-ailist *listfile*

Identifies a file containing a list of AI area files. The *listfile* file must be structured as follows:

- Specify one AI file per line
- AI file names can be fully specified absolute file name or a relative file name relative to the working directory where RFUTIL is executed

- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

`endtime`

Specifies to roll forward to a certain time. You must specify the ending time as a string of digits and separate the date and time components with a colon. Transactions are included in the partial roll forward only if they end before the specified time. For example, to roll forward to 5:10 PM on July 18, 2002, type `2002:07:18:17:10:00`. For RFUTIL to include a transaction in this partial roll forward, the transaction must have ended on or before 2002:07:18:17:09:59.

`endtrans`

Specifies to roll forward up to but not including the transaction beginning that contains the *`transaction-number`*. For example, if you specify endtrans 1000, RFUTIL rolls forward the AI area to transaction 999. If you want to include transaction 1000, you must specify endtrans 1001.

`-B` *`n`*

Specifies the number of database buffers. The single-user default value is 20.

`-r`

Indicates that buffered I/O will be used.

If the system crashes while you are running the ROLL FORWARD operation, restore your database files again and rerun the ROLL FORWARD operation.

The ROLL FORWARD qualifier displays the following information:

- The start and end dates of the AI area being applied to the database
- The number of completed transactions reapplied to the database
- The number of transactions that were active after all AI notes were applied

The ROLL FORWARD qualifier fails if:

- You omit the After-image Filename (`-a`) or AI List (`-ailist`) parameter
- It cannot open the AI area
- You name the wrong AI area
- The database was opened before all AI extents were applied

Notes

- The ROLL FORWARD qualifier always disables after-imaging for the database before beginning the roll-forward operation. After the roll-forward has completed, you must re-enable it with the AIMAGE BEGIN qualifier if you want continued AI protection.
- You must apply all AI extents associated with the database in the same sequence they were generated before you can use the database.

RFUTIL ROLL FORWARD OPLOCK qualifier

The OPLOCK qualifier provides the ability to enforce a protection mechanism during the roll forward process, preventing interruption to the target database before the final roll forward session is completed.

Syntax

```
rfutil db-name -C roll forward oplock {-a ai-name | -ailist listfile}
```

Parameters

db-name

Specifies the database you are using.

-a *ai-name*

Identifies the AI name of the specified database.

-ailist *listfile*

Identifies a file containing a list of AI area files. The *listfile* file must be structured as follows:

- Specify one AI file per line
- AI file names can be fully specified absolute file name or a relative file name relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

Note: If you specify a list of AI files with -ailist and a file is **not** an after-image file, a warning is issued and the file is skipped.

Notes

All qualifiers accepted by ROLL FORWARD are also accepted by ROLL FORWARD OPLOCK. See RFUTIL ROLL FORWARD qualifier on page 815 for details.

RFUTIL ROLL FORWARD RETRY qualifier

Restarts the roll-forward operation on an after-image extent that was in the process of rolling forward.

Syntax

```
rfutil db-name -C roll forward retry
      [ endtime yyyy:mm:dd:hh:mm:ss | endtrans transaction-number]
      [ -B n] [ -r ] [ verbose ] {-a ai-name | -ailist listfile}
```

Parameters

db-name

Specifies the database you are using.

endtime

Specifies to roll forward to a certain point in time. You must specify the ending time as a string of digits and separate the date and time components with a colon. Transactions are included in the partial roll forward only if they end before the specified time. For example, to roll forward to 5:10 PM on July 18, 2002, type 2002:07:18:17:10:00. For RFUTIL to include a transaction in this partial roll forward, the transaction must have ended on or before 2002:07:18:17:09:59.

endtrans

Specifies to roll forward up to but not including the transaction beginning that contains the *transaction-number*. For example, if you specify endtrans 1000, RFUTIL rolls forward the AI area to transaction 999. If you want to include transaction 1000, you must specify endtrans 1001.

-B *n*

Specifies the number of database buffers. The single-user default value is 20.

-r

Indicates that buffered I/O will be used.

verbose

Produces one line of information for every note in the AI area.

-a *ai-area*

Identifies the AI area of the specified database.

-ailist *listfile*

Identifies a file containing a list of AI area files. The *listfile* file must be structured as follows:

- Specify one AI file per line
- AI file names can be fully specified absolute file name or a relative file name relative to the working directory where RFUTIL is executed
- Lines that begin with a colon (:), hash mark (#), or asterisk (*) are treated as comments and skipped over for parsing

Note: If you specify a list of AI files with -ailist and a file is **not** an after-image file, a warning is issued and the file is skipped.

ROLL FORWARD RETRY enhances the support of 24 X 7 database operations. The use of this qualifier is limited to RFUTIL roll-forward operations that fail because of power outages or system failures. The ROLL FORWARD RETRY qualifier restarts the roll-forward operation on the after-image extent that was in the process of rolling forward. The retry operation finds the transaction in process at the time of failure and resumes rolling forward. It recovers the transaction log (TL) extents. Entries noting the use of the retry operation appear in the .lg file of the database. If subsequent failures occur during the retry operation, the retry operation can be restarted.

Notes

Roll forward might encounter a two-phase begin note in a BI area that will signal roll forward to enable transaction commit logging to the transaction log. If the database does not contain a TL area, roll forward will abort. To recover from this situation, you should first add a TL area to your database and then run ROLL FORWARD RETRY.

RFUTIL ROLL OPUNLOCK qualifier

Unlocks the protection enabled by ROLL FORWARD OPLOCK.

Syntax

```
rfutil db-name -C roll opunlock
```

Parameters

db-name

Specifies the database where the sequence number is being updated.

The OPUNLOCK qualifier:

- Disables the protection mechanism, making the database accessible to other users or utilities.
- Completes the phases of the crash recovery.

Notes

- The OPUNLOCK qualifier should only be used for situations with missing AI files. Once this command is run, the roll forward process stops.
- See Roll forward with the OPLOCK qualifier on page 161for more information.

RFUTIL SEQUENCE qualifier

Updates the database after-image sequence number.

Syntax

```
rfutil db-name -C sequence
```

Parameters

db-name

Specifies the database where the sequence number is being updated.

RFUTIL SEQUENCE updates the sequence number of the specified database to match the after-image extent that was active when a copy of the database was made.

Caution: RFUTIL SEQUENCE must be applied to the database prior to rolling forward any after-image extents.

Notes

RFUTIL SEQUENCE is only needed when an `AI` extent switch occurs after the source database was backed up, and before a copy is made.

28

Other Database Administration Utilities

This chapter describes miscellaneous OpenEdge database administration utilities, in alphabetical order. It discusses general parameters, and then the purpose, syntax, and primary parameters for each utility.

For details, see the following topics:

- Parameters for authentication
- DBMAN utility
- DBTOOL utility
- DSRUTIL utility
- PROADSV utility
- PROBKUP utility
- PROCLUSTER utility
- PROCOPY utility
- PRODB utility
- PRODEL utility
- PROLOG utility
- PROREST utility

Parameters for authentication

For databases operating under increased security, users must be authenticated to perform database maintenance. Additional parameters are required to authenticate the user, as follows:

`-userid` *username* [`-password` *passwd*] `-U` *username* `-P` *passwd*

A pair of username and password parameters are used together to support a common user authentication model to specify the user ID and password when connecting to an OpenEdge RDBMS.OpenEdge authenticates the user ID and password against the user accounts defined by the authentication system configured for the user's security domain.

Use username and password parameters to identify an authenticated database administrator.

If you do not specify a user ID and password for a database connection, OpenEdge assigns the OS process's user-id as the default user-id.

For more information on authentication systems and security domains, see *OpenEdge Getting Started: Identity Management*.

`-Passphrase`

For encryption-enabled databases only, specifies to prompt for a passphrase to authenticate the key store. Utilities that open the database must at a minimum authenticate as the key store user. Utilities that modify encryption policies must authenticate as the key store admin. See Transparent Data Encryption on page 199 for more information on encryption and key store authentication.

If your database is enabled for transparent data encryption and configured for manual start, you must specify `-Passphrase` every time the database is opened.

DBMAN utility

Starts, stops, or queries a database. Before you can use the DBMAN command-line utility, you must use OpenEdge Management or OpenEdge Explorer to create the database configuration and store it in the `conmgr.properties` file.

Syntax

```
dbman [ -host host-name -port port-number | service-name -user user-name ]
      -database db-name [ -config config-name -start | -stop | -query ]
```

Parameters

`-database` *db-name*

Specifies the name of the database you want to start. It must match the name of a database in the `conmgr.properties` file.

`-config` *config-name*

Specifies the name of the configuration with which you want to start the database.

`-start`

Starts the database *db-name* as defined by the configuration *config-name*.

`-stop`

Stops the database *db-name*.

`-query`

Queries the Connection Manager for the status of the database *db-name*.

`-host` *host-name*

Identifies the host machine where the AdminServer is running. The default is the local host. If your AdminServer is running on a remote host, you must use the `-host` *host-name* parameter to identify the host where the remote AdminServer is running.

`-port` *port-number* | *service-name*

Identifies the port that the AdminServer is listening on. If your AdminServer is running on a remote host, you must use the `-port` *port-number* parameter to identify the port on which the remote AdminServer is listening. The default port number is 20931.

`-user` *user-name*

If your AdminServer is running on a remote host, you must use the `-user` *user-name* parameter to supply a valid user name for that host. You will be prompted for the password.

Notes

- When you specify a user name with the `-user` parameter, Windows supports three different formats:
  - A user name as a simple text string, such as "`mary`," implies a local user whose user account is defined on the local server machine, which is the same machine that runs the AdminServer.
  - A user name as an explicit local user name, in which the user account is defined on the same machine that runs the AdminServer, except the user name explicitly references the local machine domain, for example "`\mary`".
  - A user name as a user account on a specific Windows domain. The general format is *Domain*`\`*User*, in which the *User* is a valid user account defined within the domain and the *Domain* is any valid Windows server, including the one where the AdminServer is running.
- Do not edit the `conmgr.properties` file directly. Instead, use OpenEdge Management or OpenEdge Explorer. Consult the OpenEdge Management and OpenEdge Explorer documentation or online Help for instructions.
- DBMAN supports the use of internationalization startup parameters such as, `-cpinternal` *codepage* and `-cpstream` *codepage*. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.

- The `conmgr.properties` file stores the database, configuration, and server group properties. The following is an example of the file formatting of the `conmgr.properties` file; the values are not recommended settings. Consult your OpenEdge Management and OpenEdge Explorer documentation for descriptions and details on each setting.

```
#

#
# The following are optional configuration properties and their default
# values. The legacy option, if applicable, is listed after the second
# comment. Property values set at this level become the default values
# for all configuration subgroups.
#
[configuration]
#    afterimagebuffers=5             # -aibufs
#    afterimageprocess=false         # n/a
#    afterimagestall=true            # -aistall
#    asynchronouspagewriters=1       # n/a
#    beforeimagebufferedwrites=false # -r
#    beforeimagebuffers=5            # -bibufs
#    beforeimageclusterage=60        # -G
#    beforeimagedelaywrites=3        # -Mf
#    beforeimageprocess=true         # n/a
#    blocksindatabasebuffers=0       # -B (calculated as 8*(-n))
#    casetablename=basic             # -cpcase
#    collationtable=basic            # -cpcoll
#    conversionmap=convmap.cp        # -convmap
#    crashprotection=true            # -i
#    databasecodepage=basic          # -cpdb
#    directio=false                  # -directio
#    hashtableentries=0              # -hash (calculated as (-B)/4)
#    internalcodepage=iso8859-1      # -cpinternal
#    locktableentries=10000          # -L
#    logcharacterset=iso8859-1       # -cplog
#    maxservers=4                    # -Mn
#    maxusers=20                     # -n
#    nap=1                           # -nap
#    napmax=250                      # -napmax
#    pagewritermaxbuffers=25         # -pwwmax
#    pagewriterqueuedelay=100        # -pwqdelay
#    pagewriterqueuemin=1            # -pwqmin
#    pagewriterscan=1                # -pwscan
#    pagewriterscandelay=1           # -pwsdelay
#    semaphoresets=1                 # -semsets
#    sharedmemoryoverflowsize=0      # -Mxs
#    spinlockretries=0               # -spin
#    sqlyearoffset=1950              # -yy
#    watchdogprocess=true            # n/a
#
# The following are optional database properties and their default values.
# Property values set at this level become the default values for all
# database subgroups.
#
[database]
#    autostart=false                 # autostart the defaultconfiguration?
#    databasename=demo               # absolute or relative path + database name
# The following are optional server group properties and their default
# values. The legacy option, if applicable, is listed after the second
# comment. Property values set at this level become the default values
# for all servergroup subgroups.
#
[servergroup]
#    host=localhost                  # -H
#    initialservers=0                # n/a
#    maxclientsperserver=0           # -Ma (calculated value)
#    maxdynamicport=5000             # -maxport (5000 for NT; 2000 for UNIX)
#    messagebuffersize=350           # -Mm (4gl only)
```

```
#    minclientsperserver=1           # -Mi
#    mindynamicport=3000             # -minport (3000 for NT; 1025 for UNIX)
#    networkclientsupport=true       # false for self-service
#    numberofservers=0               # -Mpb
#    port=0                          # -S ; Must be non-zero
#                                    # when networkclientsupport=true
#    prosqltrc=nnnnnnnnnnn           # turn on various levels of SQL tracing
#    reportinginterval=1             # -rpint (4gl only)
#    serverexe=<4gl server location> # _mprosrv (4gl only)
#    type=both                       # n/a

[configuration.sports2000.defaultconfiguration]
    database=sports2000
    displayname=defaultConfiguration
    servergroups=sports2000.defaultconfiguration.defaultservergroup

[database.sports2000]
    autostart=true
    configurations=sports2000.defaultconfiguration
    databasename=d:\work\database\101a\AuditEnabled\sports2000
    defaultconfiguration=sports2000.defaultconfiguration
    displayname=sports2000

[environment]

[servergroup.sports2000.defaultconfiguration.defaultservergroup]
    configuration=sports2000.defaultconfiguration
    displayname=defaultServerGroup
    port=14000
```

DBTOOL utility

Diagnostic tool that identifies possible record issues and fixes SQL Width violations.

Syntax

```
dbtool db-name
```

Parameters

db-name

Name of the database to be scanned.

When you enter the DBTOOL utility, the DBTOOL main menu appears as shown in the following figure.

Table 111: DBTOOL main menu

```
                DATABASE TOOLS MENU - 11.4
                --------------------------

          1. SQL Width & Date Scan w/Report Option
          2. SQL Width Scan w/Fix Option
          3. Record Validation
          4. Record Version Validation
          5. Read or Validate Database Blocks
          6. Record Fixup
          7. Schema Validation
          8. Disable Object Locking Protection
          9. Enable/Disable File Logging
         10. Index Space Validation
         11. Index Space Fixup
         12. Schema Fixup

          Q. Quit

          Choice:
```

The menu options provide the following functions:

- **SQL Width & Date Scan w/Report Option** — Reports on the following SQL values:
  - *_Field._Width*
  - *year_value*<0
  - *year_value*>9999
  - Array Width (applicable only to DataServers)
  - Max ArrayWidth (applicable only to DataServers)

  In the generated report, three asterisks (***) in the Error column indicate SQL width or date violations.

  See Running DBTOOL on page 828 for a description of the output generated with different logging levels.

- **SQL Width Scan w/Fix Option** — Scans for width violations over the specified percentage of the current maximum width, and increases SQL width when necessary. The SQL width calculation for OpenEdge SQL and OpenEdge DataServers are the same in all cases except for CHARACTER, RAW, and DECIMAL extent fields calculations in OpenEdge DataServers. For more information about the migration requirement differences between OpenEdge SQL and OpenEdge dataservers, see the Display prompt discussion in the next sub-section of this topic.

  In this menu option, you are prompted to enter a number indicating the percentage above the current maximum width to allocate for record growth. For example, if the current maximum width of a field is 100, and you specify 10 percent for growth, DBTOOL checks the SQL Width of the field, and if it is less than 110, increases it to 110. If SQL Width is larger than the field's current maximum width plus the percentage for growth, SQL Width is not changed.

  See Running DBTOOL on page 828 for a description of the output generated with different logging levels.

- **Record Validation** — Compares the physical storage of the record to the schema definition and reports discrepancies.
- **Record Version Validation** — Performs record validation before and after upgrading the schema version of a record. This option is used to test, through record validation, before and after upgrading the schema version of a record in memory, if a schema upgrade will succeed when an out of date record is next accessed by a client request at runtime. The first check compares the record to the current schema definition in the

record header. After that check completes, if the table has a newer definition of the schema than the record, the record schema is updated, and then the physical record is compared to the new definition.

Note that DBTOOL does not commit the records updated to the current schema version to the database, it only makes a comparison.

- **Read or validate database block(s)** — Validates the information in the database block header. There are three levels of validation:
  - Reads the block only
  - Conducts physical consistency record checks at runtime (except record overlap check)
  - Conducts physical consistency record checks at runtime

  This option can be invoked by choosing to validate either all record blocks in one area or all record blocks in all areas.

Note: The validation process will report the first error in a block then proceed to the next record block. Details of the errors encountered will be recorded in the database `.lg` file.

- **Record fixup** — Scans records for indications of possible corruption.
- **Schema Validation** — Checks for inconsistencies in the database schema. This option currently identifies errors in schema records for word indexes. If an error is detected, this option reports the index number and recommends rebuilding that index.
- **Enable/Disable File Logging** — Toggles the redirection of the tool output to a file named `dbtool.out`. DBTOOL writes this file to your current working directory.
- **Index Space Validation** — Checks for index space allocation errors.
- **Index Space Fixup** — Fixes any errors discovered with **Index Space Validation.**
- **Schema Fixup** — Fixes rare types of schema corruption. Choose one of the following:
  - **SQL View Index Cleanup** — Fixes schema corruption that occurs when there are physical indexes for SQL Views by removing these invalid indexes.
  - **_User Records Missing Fields Fixup** — Fixes a problem with databases converted using CONV1011 that have an _user table with greater than 65536 records. The _user table is missing new fields after the conversion.

  .

Notes

- DBTOOL is a multi-threaded utility. The connection type you specify when running the utility determines the number of threads DBTOOL will run. Because each thread uses a database connection, be aware of what the broker startup parameter (`-n`) is set to. Specifying 1.5*(number of CPUs) as the number of threads to run DBTOOL with is usually adequate.
- DBTOOL sends output to `.stdout`. To redirect output to a file, use the redirection symbol, as shown:

```
dbtool db-name 2 > filename.out
```

  If you decide to redirect output after entering DBTOOL, select **Enable/Disable File Logging** from the DBTOOL menu. This option redirects the output to a file named `dbtool.out`. DBTOOL writes this file to your current working directory.

- DBTOOL does not fix date errors, it only identifies them. To fix date errors, use the Data Dictionary or Data Administration tool to modify the fields that contain incompatible date values.
- The **Schema Fixup** option addresses a rare type of schema corruption that occurs when there are physical indexes for SQL Views. Physical indexes for a SQL view serve no purpose in the database and should not exist. Rarely, such an index can cause errors in database utilities performing scans, such as PROUTIL IDXBUILD. The **Schema Fixup** option removes these invalid indexes. If a database has physical indexes for SQL views, the **Schema Validation** option of DBTOOL reports these indexes as an error, as shown:

```
ERROR: Invalid _object-associate 0 for _StorageObject 2431
ERROR: _StorageObject 2431 belongs to a SQL View
Index 2431 is orphaned and should be deleted.
```

  After detecting the error, the **Schema Validation** option of DBTOOL directs you to run the **Schema Fixup** option, as shown:

```
SQL View Index errors found: 1

Please run the Schema Fixup option SQL View Index Cleanup.
```

Running DBTOOL

For most of the menu options, you must know whether or not your database has a server running. You are prompted to connect to the database as shown in the following procedure.

To run the options of DBTOOL:

1. Enter your choice at the main menu prompt.

   The following prompt appears:

```
<connect>:(0=single-user 1=self-service >1=#threads)?
```

2. Enter your connection type (`0`, `1`, or *`#threads`*).

   If a database broker is not running, you must select zero (`0`). If a database broker is running, you can run the report using one or more threads. If you enter a connection type that does not match the state of the database, DBTOOL returns this error:

```
dsmUserConnect failed rc = -1
```

   After you enter a valid connection code, the prompts specific to the functionality of your selection appear.

3. Continue through the remaining prompts for your option.

   Common prompts include the following:

- **Table** — Enter the table number at the following prompt:

```
<table>: (Table number or all)?
```

  You may be prompted to further refine your selection:

  - If the table you select is a multi-tenant table, you will be prompted with the following sub-prompt:

```
<multi-tenant>:(tenant tname | group gname | shared | all)?
```

    If you specify `tenant` or `group`, dbtool only evaluates the partition owned by the named tenant or group. If you specify `shared`, the shared partition is evaluated. Specifying **`all`** evaluates the entire table.

  - If the table you select is a table partitioned by table-partitioning, you will be prompted with the following sub-prompt:

```
<Partition Name>: (Partition name or all)?
```

    If you specify a partition name, dbtool only evaluates the named table partition. Specifying **`all`** evaluates the entire table.

- **Area** — Enter the area number at the following prompt:

```
<area>: (Area number or all)?
```

  If the area you select contains multi-tenant tables or indexes, you will be prompted with the following sub-prompt:

```
 <tenant/group>:(tenant tname | group gname | shared | all)?
```

  Refine the data that dbtool evaluates. If you specify `tenant` or `group`, dbtool only evaluates the partitions owned by the named tenant or group. If you specify `shared`, the shared partition is evaluated. Specifying **`all`** evaluates the entire area.

- **Display** — Enter the verbose level. The verbose level defines the amount of output displayed, with zero being the least verbose. The following prompt appears for display:

```
<display>: (verbose level 0-4)?
```

  The SQL Width Scan options are used to display or set the width of columns based on the size of the actual data in the fields. These calculations are sized in the same way for OpenEdge SQL and OpenEdge DataServers except in the case of CHARACTER, RAW, and DECIMAL extent columns in OpenEdge DataServers. For information on the SQL Width & Date Scan w/Report Option, see DBTOOL utility on page 825.

When the SQL Width Scan options are applied to OpenEdge SQL, use verbose levels 0-3 to display information about column width calculations made by the tool. For OpenEdge SQL, the "_Width" value for the extent fields is the total of the data size of each individual element of the extent added together. The verbose levels report varying levels of detail, depending on the option selected, but generally follow the model:

- **Level 0** — Report summary information
- **Level 1** — Report errors for each record
- **Level 2** — Report record size and the number of fragments, errors etc
- **Level 3** — Report on binary data in the records
- **Level 4** — Report details for OpenEdge DataServer only

  When the SQL Width Scan options are applied to OpenEdge DataServers, use verbose level 4 to display information about column width calculations along with the **Array Width** and **Max ArrayWidth** information. These additional display fields (Array Width and Max ArrayWidth) show column widths that are adjusted differently for CHARACTER, RAW, and DECIMAL extent fields as needed to support DataServers and foreign data sources. The "_Width" schema field is not changed directly. Therefore, OpenEdge SQL users are not affected by the alternative calculations for DataServer extent fields.

  The **SQL Width Scan w/Fix Option** stores a hidden schema field containing the width value required by the OpenEdge DataServers. The calculated extent width identifies the largest single data element in the array and multiplies its length with the number of elements in the extent. This calculation ensures that an adequate column width is derived for the columns migrating to a foreign data source. At the time of migration, the hidden schema field is compared to the current width value in the schema, and the foreign database column size is based on the larger of the two. The width value derived for the extent field when the file was originally added to the schema is based on the field format.

Note: The hidden schema field for the OpenEdge DataServers is set irrespective of the verbose level selected.

DSRUTIL utility

For databases running OpenEdge Replication, a utility to perform OpenEdge Replication server, OpenEdge Replication agent, source database, and target database requests.

Syntax

```
DSRUTIL db-name -C ACTION
     [Server | Agent] [name | ALL] [ -Passphrase ]
```

See *OpenEdge Replication: User Guide* for details on using DSRUTIL.

PROADSV utility

Starts, stops, or queries the current installation of an AdminServer on UNIX.

Syntax

```
proadsv { -start | -stop | -query }
   [ -port port-number] [ -adminport port-number]
   [ -cluster ] [ -hostname host-name]
   [ -help ]
```

Parameters

`-start`

Starts the AdminServer.

`-stop`

Stops the AdminServer.

`-query`

Displays AdminServer status.

`-port` *port-number*

Specifies the listening port number for online command utilities, such as DBMAN. If a port number is not specified, it defaults to 20931.

`-adminport` *port-number*

Specifies the listening port number for communication between a server group and an AdminServer. The default port number is 7844.

`-agentport` *port-number*

Specifies the port number used by the database management agent to communicate with the AdminServer.

To override the default value, specify the value in the command prompt or change it in the agent.properties file.

`-mgmtport` *port-number*

Specifies the port number used by the AdminServer to accept remote monitoring connections from OpenEdge Management. This defaults to 6835.

To avoid conflicts with other AdminServers running on the same machine, override the default value.

`-cluster`

Specifies to the AdminServer that it is running in a clustered environment.

`-hostname` *host-name*

Specifies that the AdminServer should bind the RMI registry to the host *host-name*. In a clustered environment, use the cluster alias as the value for *host-name*. In the event of a failover, using the cluster alias guarantees availability of the RMI registry for the AdminServer.

`-help`

Displays command-line help.

Notes

- In Windows, the AdminServer runs as a service. The AdminServer is configured to automatically start. You can change the listening port for the AdminServer by adding `-port`*`port-number`* or `-adminport`*`port-number`* (either one or both) to the following registry keys:

```
HKEY_LOCAL_MACHINE\SOFTWARE\PSC\AdminService\version\StartupCmd

or

HKEY_LOCAL_MACHINE\SOFTWARE\PSC\AdminService\version\ShutdownCmd
```

  To change the default port, add `-port` or `-adminport` and the port number to the end of the value. If you add both `-port` and `-adminport`, be sure not to use the same port number. Be sure to leave a space between `-port` or `-adminport` and the port number. For example:

```
....AdminServer -port 9999 -adminport 7844
```

- To run more than one AdminServer on a single system, specify a unique `-adminport` *`port-number`* and `-port` *`port-number`* for each AdminServer. Failure to do so can result in communication errors between AdminServer and server groups.
- An AdminServer is installed on every system where you install an OpenEdge database, ORACLE DataServer, AppServer, NameServer, WebSpeed® Transaction Server, or WebSpeed Messenger. The AdminServer must be running in order to use OpenEdge Management, OpenEdge Explorer, or any of the command-line configuration utilities, such as DBMAN.
- See Starting Up and Shutting Down on page 73 for additional information describing the Unified Broker Framework, including an AdminServer.
- PROADSV supports the use of internationalization startup parameters such as `-cpinternal` *`codepage`* and `-cpstream` *`codepage`*. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.

PROBKUP utility

Backs up an OpenEdge RDBMS, including the database, before-image files, and transaction log (TL) extents, and optionally enables after-imaging and AI File Management during an online backup.

Syntax

```
probkup [ online ] db-name [ incremental ] device-name
        [ enableai [-aiencryption enable | disable ] ]
        [ enableaiarchiver -aiarcdir dirlist
            [-aiarcinterval n] [-aiarcdircreate] ]
        [ bibackup { active | all } ]
        [   -estimate | -vs n | -bf n | -verbose | -scan
          | -io i | -com | -red i | -norecover
        ] . . .
```

Parameters

`online`

Indicates the backup is an online backup.

`db-name`

Specifies the database you want to back up.

`incremental`

Indicates the backup is an incremental backup.

`device-name`

Identifies a special device (for example, a tape drive) or a standard file. If *`device-name`* identifies a special device, PROBKUP assumes the device has removable media, such as a tape. For Windows, use `\\.\tape0` for the device name if you are backing up to a tape drive.

`enableai`

Directs PROBKUP to enable after-imaging as part of the online backup process. The backup must be full and online to enable after-imaging. See After-imaging on page 139 for more information on enabling after-imaging online.

`-aiencryption [ enable | disable ]`

If you are enabling after-imaging on an encryption-enabled database, `-aiencryption` explicitly enables or disables encryption of your AI files. If not specified, your AI files are encrypted or decrypted based on prior settings. You can determine the encryption status of your AI files with PROUTIL DESCRIBE; see PROUTIL DESCRIBE qualifier on page 678 for more information. See Transparent Data Encryption on page 199 for more information on encryption.

`enableaiarchiver -aiarcdir` *`dirlist`*

Directs PROBKUP to enable the AI File Management Utility as part of the online backup process. You must supply *`dirlist`*, a comma separated list of directories where archived after-image files are written by the AI File Management Utility. The directory names can not have any embedded spaces. In Windows, if you specify more than one directory, place *`dirlist`* in quotes. The directories

must exist, unless you also specify `-aiarcdircreate` to direct the utility to create the directories. For more information, see AI File Management utility on page 150.

`-aiarcdircreate`

Directs the AI File Management utility to create the directories specified by `-aiarcdir`.

`-aiarcinterval` *n*

Include this parameter to specify timed mode operation of AI File Management. Omit this parameter for on-demand mode, and no mandatory extent switches other than when an extent is full. *n* sets the elapsed time in seconds between forced AI extent switches. The minimum time is 2 minutes, the maximum is 24 hours.

`bibackup { active | all }`

For online backup ONLY, indicates whether to backup the `active` BI clusters or `all` the BI clusters. By default, only the active clusters are backed up.

If you do not specify `bibackup all`, online backup is not backward compatible to releases prior to OpenEdge Release 11.3.0.

`-estimate`

Indicates that the backup will give a media estimate only. Use the Scan parameter when using the Incremental or Compression parameters to get an accurate estimate.

Use `-estimate` for offline backups only. PROBKUP does not perform a backup when the `-estimate` parameter is used.

`-vs` *n*

Indicates the volume size in database blocks that can be written to each removable volume. Before PROBKUP writes each volume, it displays a message that tells you to prepare the next volume. After writing each volume, a message tells you to remove the volume.

If you use the Volume Size parameter, the value must be greater than the value of the Blocking Factor parameter.

If you do not use `-vs`, PROBKUP assumes there is no limit and writes the backup until completion or until the volume is full. When the volume is full, PROBKUP prompts you for the next volume.

`-bf` *n*

Indicates the blocking factor for blocking data output to the backup device. The blocking factor specifies how many blocks of data are buffered before being transferred to the backup device. The primary use for this parameter is to improve the transfer speed to tape-backup devices by specifying that the data is transferred in amounts optimal for the particular backup device. The default for the blocking factor parameter is 34. The maximum value is 1024.

`-verbose`

Directs the PROBKUP utility to display information during the backup. If you specify the Verbose parameter, PROBKUP displays "Backed up *n* blocks in `hh:mm:ss`" every 10 seconds. If you do not specify the Verbose parameter, the message appears only once when the backup is complete.

`-scan`

Directs PROBKUP to perform an initial scan of the database and to display the number of blocks that will be backed up and the amount of media the backup requires. You cannot use the `-scan` parameter for online backups.

For full backups, if you specify **`-scan`** as well as `-com`, PROBKUP scans the database and computes the backup media requirements after the data is compressed.

`-io` *i*

Specifies an incremental overlap factor. The incremental overlap factor determines the redundancy among incremental backups. An incremental overlap factor of one (1) on every backup allows for the loss of one incremental backup in a backup series, as long as the immediate predecessor of that backup is not also lost. An overlap factor of two (2) allows for losing the two immediate predecessors. The default overlap factor is zero(0).

`-com`

Indicates that the data should be compressed prior to writing it on the backup media. The unused portion of index and record blocks is compressed to a 3-byte compression string. Free blocks are compressed to the length of their header, 16 bytes.

`-red` *i*

Sets the amount of redundancy to add to the backup for error correction. The value *i* is a positive integer that indicates the number of blocks for every error correction block. PROBKUP creates an error correction block for every *i* blocks and writes it to the backup media. You can use error correction blocks to recover corrupted backup blocks. See Backing Up a Database on page 97 for more information about error correction blocks and data recovery.

The lower the redundancy factor, the more error correction blocks are created. If you specify a redundancy of one (1), you completely duplicate the backup, block for block. Because of the amount of time and media required to create the error correction blocks, use this parameter only if your backup media is unreliable. The default for the redundancy parameter is zero (0) indicating no redundancy.

`-norecover`

Do not perform crash recovery before backing up the database, but back up the BI files.

Notes

- When restoring a backup, the target database must contain the same physical structure as the backup version. For example, it must have the same number of storage areas, records, blocks, and blocksize.
- The minimum backup volume size is 34K.
- You cannot perform an online backup on:
  - A system running in single-user mode
  - A database that was started with the No Shared Memory (`-noshm`) parameter
  - A database that was started with the No Crash Protection (`-i`) parameter
- If after-imaging is enabled prior to this online backup, PROBKUP automatically switches to a new after-image (AI) file before starting the backup process, establishing a reference point from which to start your roll-forward recovery. Before you perform an online backup, verify that the next AI file is empty. If the next AI file is not empty, PROBKUP aborts the backup. See After-imaging on page 139 for more information about AI files.

- If you run the PROBKUP utility at the same time another process is accessing the same backup device, you might receive a sharing violation error.
- If you use the Compression parameter (`-com`), you reduce the size of your backup by between 10 and 40 percent, depending on your database.
- Incremental backups vary in size according to the amount of changes made between incremental backups. Incremental backups that specify an Overlap (`-io`) value, also vary in size, depending on the amount of changes accumulated during the entire overlapping backup period. See Incremental backups on page 91 for more information about incremental backups.
- If the BI file is not truncated before you perform a backup, the database engine performs database recovery.
- Enabling After-Image Management requires after-imaging to be enabled. Both can be enabled simultaneously or after-imaging can be previously enabled.
- Enabling After-Image Management directs the broker to start the AI Management daemon.
- See Backing Up a Database on page 97 for more information about performing backups.
- PROBKUP supports the use of internationalization startup parameters such as `-cpinternal codepage` and `-cpstream codepage`. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.

Note: The number of times a database featuring Type I storage areas may backed up using the PROBKUP utility is limited for databases operating on OpenEdge Release 11.7.5 or earlier. This limit is tracked by the Backup Counter Increment (bk_incr) reported in the Master Block. Exceeding this limit will result in a Backup Counter overflow and a corrupted database backup file.

The Backup Counter Increment limit is 65,535. In applicable terms, this limit amounts to approximately 22 months for a database that is backed up every 15 minutes. For a database that is backed up every hour, this limit would be reached in approximately 7.5 years.

To reset the Backup Counter Increment, a full dump and load of the database must be performed.

The limit for the Backup Counter Increment is higher in Type II storage areas. To avoid a Backup Counter overflow, Progress recommends moving all user data from the Schema area and Type I areas into Type II areas.

As of OpenEdge Release 12, the Backup Counter Increment has been increased, and users are unlikely to experience overflow. However, users performing a full online backup of databases featuring Type I storage areas may experience increased wait times. Moving data from Type I areas into Type II areas will eliminate any extended wait times. Performing more frequent incremental backups will also decrease these wait times.

PROCLUSTER utility

The PROCLUSTER command-line interface provides a user-friendly interface to Clusters. With PROCLUSTER, you can:

- Enable a database for failover
- Disable a cluster-enabled database
- Start and stop a cluster-enabled database
- Query the status of a cluster-enabled database

Syntax

```
procluster db-name
  [enable [-pf params-file] [AI] [BI] [APW=n] [WDOG]
 |disable |start |stop |terminate |  looksalive |isalaive ]
```

Parameters

db-name

Specifies the fully qualified path of the database.

enable

Enables a database so that it fails over properly. Before the database can be enabled, it must be created and reside on a shared disk. The database must not be in use when it is enabled as a cluster resource.

When you enable a database as a resource using PROCLUSTER enable, Clusters performs the following:

- Examines the database structure and generates a registration file
- Logs the registration in the database's log file
- Performs the cluster-specific registration

Enabling an OpenEdge database as a cluster resource creates the file *<dbname>*Resources.cluster. This is a text file that contains names of resources dependent on the database resource. If the database was not enabled with any helper processes or performance enhancers like AIW, APW, BIW, WDOG, the .cluster file will be empty; otherwise, it will contain the resource names of the dependent resources as registered with the cluster manager. **Do not** edit or delete this file. Clusters manages this file, and deletes it when the database is disabled.

If PROCLUSTER enable is successful, it returns the following message:

```
The cluster REGISTER command was successful. (10532)
```

-pf *params-file*

Specifies the file containing any parameters that the database requires when started.

The parameter file is required to:

- Be named *db-name*.pf
- Reside in the same directory as the database .db file
- Contain the parameter -cluster protected
- For Solaris only, the length of the mount point of the database cannot exceed 18 characters

AI

Directs PROCLUSTER to enable the after-image files, and an AI writer as resources with a dependency on the database. After-imaging must have previously been enabled on the database. See After-imaging on page 139 for information on after-imaging.

`BI`

Directs PROCLUSTER to enable a BI writer as a resource with a dependency on the database.

`APW=` *`n`*

Directs PROCLUSTER to enable *`n`* number of asynchronous page writers as a resource with a dependency on the database.

`WDOG`

Directs PROCLUSTER to enable the Watchdog process as a resource with a dependency on the database.

`disable`

Removes the identified database resource if the database has been previously enabled for failover. Specifying the database name automatically disables any other optional dependencies specified when the database was enabled.

When you remove a database resource using PROCLUSTER disable, Clusters does the following:

- Shuts down the database if it is running
- Deletes the resource from the cluster manager software once it is in an offline state

Deletes the group from the cluster manager software if the resource is the last resource in the resource group.

`start`

Starts a cluster-enabled database. The database must have been previously enabled a cluster resource.

Note: PROCLUSTER will append `-pf`*`db-name`*`.pf` to the `proserve` command that is generated to start the database. The start command will fail if this parameter file is not found.

`stop`

Stops a cluster-protected database. The database must be a cluster resource. When you stop the database with PROCLUSTER stop, Clusters does the following:

- Stops the database
- Notifies the cluster that the resource should be stopped without fail over

`terminate`

Forces a registered or cluster-protected database to shut down. The database must be a cluster resource.

`looksalive`

Determines if a resource *looks operational* by querying the cluster manager cache, if available, or the active system for the status of the specified database. The database must be a cluster resource.

The `looksalive` exit status returns the following text if the database looks alive:

```
Resource: db-name State: Looks Alive
```

The only state of the database that returns a value of Looks Alive is when the database is enabled and started. All other states will return the following:

```
Resource: db-name State: Not Alive
```

`isaliave`

Determines if a resource is *actually operational* by querying the active system for the status of the specified database.

The `isalive` exit status returns the following text if the database returns a successful query:

```
Resource: db-name State: Is Alive
```

All other states return the following:

```
Resource: db-name State: Not Alive
```

Notes

- The PROCLUSTER ENABLE registers the database as a cluster resource, even if there are errors in the command for the helper processes. To correct the errors, you must first use PROCLUSTER DISABLE to unregister the database, then use PROCLUSTER ENABLE to re-register without errors.
- If any of the helper (APW, BIW, or AIW) processes or the watchdog (WDOG) process specified to start with a particular database when the PROCLUSTER START command is issued fails, then they will not be started on another machine during a failover. They produce an OFFLINE status.

PROCOPY utility

Copies an existing database.

Syntax

```
procopy source-db-name target-db-name [-newinstance] [-silent] [-relative]
```

Parameters

source-db-name

Specifies the database you want to copy. You cannot copy the database if you have a server running on it.

target-db-name

Specifies the structure file or the new database. If you specify a directory without a filename, PROCOPY returns an error.

The value you specify can be any combination of letters and numbers, starting with a letter. Do not use ABL keywords or special characters, such as commas or semicolons. The maximum length of *target-db-name* varies depending on the underlying operating system. For specific limits, see your operating system documentation.

-newinstance

Specifies that a new GUID be created for the target database.

-silent

Suppresses the output of work-in-progress messages.

-relative

If your source database is a relative-path database, or if you have an existing relative-path `.st` file for your target, create the target database as a relative-path database. By default, copying a database creates a target with absolute path names.

PROCOPY copies a source database to a target database. If the target database exists, it must contain at a minimum the same type and number of storage areas and same extent types as the source database. However, the number of extents in the storage areas of the target database do not need to match the number of extents in the source database. PROCOPY will extend the existing extents in the target database to accommodate the possible increase in size.

If a target database does not exist, PROCOPY creates one using an existing `.st` file in the target database directory. If a `.st` file does not exist, PROCOPY creates the target database using the structure of the source database and places all of the extents in the same directory as the target database `.db` file, even when the source database resides in multiple directories.

Notes

- If you do not supply the `.db` extension for databases, PROCOPY automatically appends it.
- A target database must contain the same physical structure as the source. For example, it must have the same number of storage areas, records, blocks, blocksize, and cluster size.
- PROCOPY supports the use of internationalization startup parameters such as `-cpinternal codepage` and `-cpstream codepage`. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.
- Databases supplied in the install directory for this release, contain the field for a database GUID, but the field contains the `Unknown value (?)` rather than a valid GUID. When PROCOPY is used to copy one of these databases, the GUID field of the target database is automatically set.

PRODB utility

Creates a new OpenEdge database.

Syntax

```
prodb [ new-db-name ]
      [ empty | sports | isports | sports2000 | old-db-name | demo ]
      [ -newinstance ]
```

Parameters

new-db-name

Specifies the name of the database you are creating. If you specify a directory without a filename, PRODB returns an error.

The value you specify can be any combination of letters and numbers, starting with a letter. Do not use ABL keywords or special characters, such as commas or semicolons. The maximum length of *new-db-name* varies, depending on the underlying operating system. See OpenEdge RDBMS Limits on page 59 for details on specific limits.

`empty`

Specifies that the new database is a copy of the `empty` database located in the OpenEdge install directory. PRODB knows where to locate the `empty` database, so you do not need to provide a pathname to it.

In addition to the default `empty` database, PRODB allows you to create other empty database structures with different block sizes:

- empty (default).
- empty1 (1K block size).
- empty2 (2K block size).
- empty4 (4K block size).
- empty8 (8K block size).

To create these `empty` database structures, however, you must specify the pathname to where OpenEdge is installed, or use the DLC environment variable. For example, use the following command:

```
prodb new-db-name $DLC/empty2
```

`sports`

Specifies that the new database is a copy of the Sports database.

`isports`

Specifies that the new database is a copy of the international Sports database.

`sports2000`

Specifies that the new database is a copy of the Sports2000 database.

`old-db-name`

Specifies the name of the database you are copying.

`demo`

Specifies that the new database is a copy of the demo database.

`-newinstance`

Specifies that a new GUID be created for the target database.

PRODB creates a new database from a specified source database. PRODB creates a new database using the structure of the source database and places all of the extents in the current working directory. You can use PRODB to make a copy of any of the demonstration or empty databases.

Notes

- You can also create a new database from the Data Dictionary. See the Data Dictionary online Help, or *OpenEdge Development: Basic Database Tools* for more information.
- When you use the PRODB utility and give the copy a new name, you cannot run the original r-code against the new database. This is because PRODB saves the database with the r-code. To run the r-code against the new database, use the Logical Database Name (`-ld`) startup parameter and use the original database name.
- A new database must contain the same physical structure as the source. For example, it must have the same number of storage areas, records, blocks, and blocksize.
- Databases supplied in the install directory for this release, contain the field for a database GUID, but the field contains the `Unknown value (?)` rather than a valid GUID. When PRODB is used to copy one of these databases, the GUID field of the target database is automatically set.
- PRODB supports the use of internationalization startup parameters such as `-cpinternal codepage` and `-cpstream codepage`. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.
- When issuing the PRODB command, specify the target and source database names (they are positional) before specifying an internationalization startup parameter such as `-cpinternal`.
- PRODB also supports the parameter file `(-pf)` startup parameter that contains a list of valid startup parameters.

PRODEL utility

Deletes an OpenEdge database.

Syntax

```
prodel db-name
```

Parameters

db-name

Specifies the database you are deleting.

Notes

- When you delete a database, PRODEL displays a message to notify you that it is deleting the database, log, BI, and AI areas for that database.
- When you delete a database, PRODEL deletes all associated areas (database, log, before-image (BI) and after-image (AI) files) that were created using the structure description file.
- If an AI area exists in a database, a warning appears. Back up the AI area before deleting it.
- PRODEL supports the use of internationalization startup parameters such as `-cpinternal codepage` and `-cpstream codepage`. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.
- PRODEL opens the database file to delete it.
- PRODEL does not delete the database's `.st` file. Because the `.st` file remains, you can recover the database after it has been deleted.

PROLOG utility

Truncates the log file.

Syntax

```
prolog db-name [ -online ] [ -silent ]
```

Parameters

db-name

Specifies the database log you want to truncate. Do not include the `.db` suffix.

`-online`

Optionally specifies that the database whose log is being truncated is currently online. PROLOG truncates the log file of an online database with or without specifying `-online`. The parameter is maintained for backward compatibility.

`-silent`

By default when your database is online, PROLOG re-logs all your database startup parameters to the log file. Specify `-silent` if you do not want to log the startup parameters.

The database log (`.lg`) file expands as you use the database. If it becomes too large, use the `PROLOG` utility, in either an offline or online mode, to reduce the event log file's size.

- **Offline** — PROLOG removes old log entries, preserving approximately the last 3072 bytes of the file.

- **Online** — Removes entries in the database log file, exactly as the offline mode, but then re-logs the database startup parameters to the log file, if `-silent` is not specified. It is possible to lose some messages while performing the online truncation.

Notes

- If you want to save the log file, use operating system utilities to back it up before using PROLOG.
- See Logged Data on page 439 for more information about log files.
- PROLOG supports the use of internationalization startup parameters such as `-cpinternal` *codepage* and `-cpstream` *codepage*. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.

PROREST utility

Verifies the integrity of a database backup or restores a full or incremental backup of a database.

Syntax

```
prorest db-namedevice-name
          { -vp | -vf | -list }
          [ -dbrestrict restrict-type ] [ -newinstance ]
          [-verbose ]
```

Parameters

db-name

Specifies the database where you want to restore the backups.

device-name

Identifies the directory pathname of the input device (for example, a tape drive) or a standard file from which you are restoring the data. If *device-name* identifies a block or character special device, PROREST assumes the device has removable media, such as a tape or a floppy diskette.

`-vp`

Specifies that the restore utility reads the backup volumes and computes and compares the backup block cyclic redundancy checks (CRCs) with those in the block headers. To recover any data from a bad block, you must have specified a redundancy factor when you performed the database backup. See Backing Up a Database on page 97 for more information about error correction blocks and data recovery.

`-vf`

Specifies that the restore utility only compares the backup to the database block for block.

`-list`

Provides a description of all application data storage areas contained within a database backup. Use the information to create a new structure description (`.st`) file and database so you can restore the backup.

`-dbrestrict` *`restrict-type`*

Protects the restored database from updates by other users by putting the database in a restricted access mode where *`restrict-type`* is one of the following values:

- **`datamove`** — Indicating that the next allowable activity on this database is moving data with the datamove utility
- **`partitioncopy`** — Indicating that the next allowable activity on this database is copying data with the partitionmanage copy utility

Adding `-dbrestrict` enables the same restrictions as PROUTIL DBRESTRICT. See PROUTIL DBRESTRICT qualifier on page 675 for details of the restrictions.

`-newinstance`

Specifies that a new GUID be created for the target database.

`-verbose`

Displays additional information as the restore is performed, such as:

- If the target of the restore doesn't exist, `-verbose` lists the target area size information.
- For restore full verify (`-vf`) and restore partial verify (`-vp`), `-verbose` displays periodic updates that include the number of blocks verified, the elapsed time, and an estimate of the time remaining given the current rate of progress.

Notes

- Restore a backup with the same OpenEdge release that you used to perform the backup. In most cases, you can restore a backup with any minor release within the same major release, with the following exceptions:
  - If you enabled a feature, you cannot restore your database to a release that does not support that feature, without first disabling the feature from your database. For example, if your database has Change Data Capture enabled, you cannot restore a backup of that database to an OpenEdge release that does not support Change Data Capture. Be aware that New VST tables are enabled by default starting in Release 11.5, and are a feature under this restriction. Use PROUTIL DESCRIBE to list the features enabled for your database.
  - For online backup ONLY, if you do not specify “bibackup all”, online backup is not backward compatible to releases prior to OpenEdge Release 11.3.0. The “bibackup” qualifier indicates whether to backup the active BI clusters or all the BI clusters. By default, only the active clusters are backed up.
- When restoring a backup, the target database must contain the same physical structure as the backup version. For example, it must have the same number of storage areas, records per block, and block size.
- For systems running with an Enterprise license, performing PROREST on an existing database will result in a target database with large file support enabled. For systems running with a sub-Enterprise license, performing PROREST on an existing database will result in a target database with large file support disabled.

Note: PROREST does not ensure that the target database is sized to hold all the restored data. In the event that the target database is too small, the behavior of PROREST is unchanged, it exits with an error.

- The -keeptargetlfe parameter is no longer supported. Running PROREST with -keeptargetlfe will no longer have any effect on the resulting database as it did in releases prior to OpenEdge Release 12.0.
- PROREST creates storage areas that match the information provided by the -list qualifier. However, if you do not create and apply a structure definition (.st) file to your target database, PROREST creates one variable-length extent per area. To have multiple extents, you must create and apply a structure definition file to your target database before performing the restore.

  The following figure shows an example of the output generated by the -list qualifier.

  Table 112: Sample PROREST -list output

```
OpenEdge Release 10.2B1P as of Wed Oct 21 19:01:48 EDT 2009
Area Name: Schema Area
       Size: 11264, Records/Block: 32, Area Number: 6, Cluster Size: 1
Area Name: Info Area
       Size: 1024, Records/Block: 32, Area Number: 7, Cluster Size: 1
Area Name: Customer/Order Area
       Size: 6656, Records/Block: 32, Area Number: 8, Cluster Size: 8
Area Name: Primary Index Area
       Size: 112, Records/Block: 1, Area Number: 9, Cluster Size: 8
Area Name: Customer Index Area
       Size: 256, Records/Block: 1, Area Number: 10, Cluster Size: 64
Area Name: Order Index Area
       Size: 8192, Records/Block: 32, Area Number: 11, Cluster Size: 64
Area Name: Encryption Policy Area
       Size: 20448, Records/Block: 32, Area Number: 12, Cluster Size: 64
Area Name: Audit Area
       Size: 4608, Records/Block: 32, Area Number: 20, Cluster Size: 8
Area Name: Audit Index
       Size: 8704, Records/Block: 32, Area Number: 22, Cluster Size: 8
```

  Use the output of PROREST -list to calculate the size of each restored area as follows:

```
area-size = (Size / records-per-block) * database-block-size
```

  For example, the size of the restored schema area is:

```
area-size = (Size / records-per-block) * database-block-size
1,441,792 = (11264 / 32) * 4096
```

  If you do not know your database block size, the output of PROREST -vp or -vf provides the information.

- PROREST restores transaction log (TL) extents.
- If your database is enabled for transparent data encryption, you must restore your key store file (*dbname*.ks). You must back up and restore your key store independent of your database using operating system utilities. The key store is not backed up by PROBKUP or restored by PROREST.
- Before you restore a database, you might want to verify that your backup does not contain any corrupted blocks. You can use the PROREST utility to verify the integrity of a full or incremental backup of a database by using the Partial Verify or Full Verify parameters.
- The Partial Verify or Full Verify parameters do not restore or alter the database. You must use the PROREST utility separately to restore the database.
- You can use the Partial Verify parameter with both online and offline backups.

- Use the Full Verify parameter immediately after performing an offline backup to verify that the backup is correct.
- PROREST supports the use of internationalization startup parameters such as `-cpinternal`*`codepage`* and `-cpstream`*`codepage`*. See Database Startup Parameters on page 473 for a description of each database-related internationalization startup parameter.

29

SQL Utilities

This chapter describes the OpenEdge database administration utilities for SQL.

For details, see the following topics:

- SQLDUMP utility
- SQLLOAD utility
- SQLSCHEMA utility

SQLDUMP utility

A command-line utility that dumps application data from SQL tables into one or more files.

Syntax

```
sqldump -u user_name [ -a password] [ -C code_page_name]
        -t [owner_name.] table_name1 [ [,owner_name.] table_name2,...]
        [ -n tenant_name1,...]
        db_name
```

Parameters

`-u user_name`

Specifies the user id SQLDUMP used to connect to the database. If you omit the `user_name` and `password` parameter values, SQLDUMP prompts you for the values. If you omit `user_name` and supply a `password`, SQLDUMP uses the value defined in the USER environment variable as the user_name value.

`-a password`

Specifies the `password` used by the database for authentication.

`-C code_page_name`

A case-insensitive character string that specifies the name of the dump file's code page. If the `-C` parameter specifies a code page name that is not valid, the utility reports a run-time error. If the `-C` parameter does not appear at all, the code page name defaults to the client's internal code page:

- If set, the value of the client's SQL_CLIENT_CHARSET environment variable
- If not set, the name of the code page of the client's locale

For example, you might use the `-C` parameter to have a Windows client using the MS1250 code page produce a dump file using the ISO8859-2 code page (to read later on a UNIX machine, perhaps). Although you can accomplish this by setting the client's SQL_CLIENT_CHARSET environment variable, using the `-C` parameter might be easier.

`-t owner_name.table_name`

Specifies a list of one or more tables to dump to a file. This parameter is required. Pattern matching is supported in both *owner_name* and *table_name*, using a percent sign (%) for one or more characters and an underscore (_) for a single character. The pattern matching follows the standard defined by the `LIKE` predicate in SQL.

You can dump a single table, a set of tables, or all tables. If you omit the optional *owner_name* qualifier, SQLDUMP uses the name specified by the `-u` parameter.

`-n tenant_name`

Specifies a list of one or more tenants to dump to a file. This parameter is optional.

db_name

Specifies the database where you are dumping tables. You can dump tables from one database each time you invoke SQLDUMP. There is no option flag preceding the *db_name*. This parameter is required and must be the last parameter specified. The database name is specified in the following way: `progress:T:`*localhost*`:`*demosv*`:`*jo*`.`

SQLDUMP dumps application data from SQL tables into one or more files. You can load the data from the files into another database with the SQLLOAD utility. The SQLDUMP utility does not dump data from ABL tables.

The SQLDUMP utility writes user data in row order into ASCII records with variable-length format. The column order in the files is identical to the column order in the tables. The utility writes both format and content header records to the dump file. You can dump multiple tables in a single execution by specifying multiple table names, separated by commas. Make sure there are no spaces before or after commas in the table list.

Data for one table always goes to a single dump file. Each dump file corresponds to one database table. For example, if you specify 200 tables in the SQLDUMP command, you will create 200 dump files. The SQLDUMP utility assigns the filenames that correspond to the *owner_name* and *table_name* in the database, with the file extension `.dsql`. If a dump file for a specified table already exists, it will be overwritten and replaced. Dump files are created in the current working directory.

The format of the records in a dump file is similar to the ABL `.d` file format:

- Converts all values to character representation
- Delimits CHARACTER values with double quotes
- Can contain any embedded characters except for NULL values, allowing commas, newlines, and other control characters
- Uses two sets of double quotes to escape embedded double quotes
- Delimits NUMERIC and other non-character data types using a space
- Processes TIMESTAMP data as if it were CHARACTER data
- Has a size limit of 2K for a single column value
- Has a maximum record length of 32K for dump file records

Any error is a fatal error, and SQLDUMP halts the dumping process so that data integrity will not be compromised. SQLDUMP reports errors to standard output.

After successful processing, SQLDUMP writes a summary report to standard output. For each table SQLDUMP processes, the report shows:

- Table name
- Dump filename
- Number of records dumped
- Number of bytes dumped
- Number of seconds required for processing

Example

This example directs the SQLDUMP utility to write the data from two tables to two dump files. The *user_name* and *password* for connecting to the database are `tucker` and `sulky`. The `tucker` account must have the authority to access the `customers` and `products` tables in database `salesdb` with *owner_name*`martin`, as shown:

```
sqldump -u tucker -a sulky -t martin.customers,martin.products
progress:T:thunder:4077:salesdb
```

This example directs the SQLDUMP utility to write the data from all tables in the `salesdb` database that begin with any of these strings: `cust`, `invent`, and `sales`, and having any owner name that the user `tucker` has authority to access. The *user_name* and *password* for connecting to the database are `tucker` and `sulky`, as shown:

```
sqldump -u tucker -a sulky -t%.cust%,%.invent%,%.sales%
progress:T:thunder:4077:salesdb
```

This example directs the SQLDUMP utility to write the data from all tables for all owner names in the salesdb database:

```
sqldump -u tucker -a sulky -t %.%  progress:T:thunder:4077:salesdb
```

This example directs the SQLDUMP utility to dump the data from the tenants ten1 and ten2 to two SQL dump files respectively. The *user_name* and *password* to connect to the database are supertenUser@superdom and superten. The supertenUser account in the superdom domain must have the authority to access the ten1 and ten2 tenant tables in mtdb database.

To separate the tenant specific data, SQLDUMPutility creates separate directories for each tenant. The ten1/<OWNER>.MTTAB1.DSQL and ten2/<OWNER>.MTTAB1.DSQL are the two directories that are created to dump tenant data:

```
sqldump -u supertenUser@superdom -a superten  -t mttab1 -n ten1, ten2
progress:T:localhost:9999:mtdb
```

If regTenantUser is mapped to a regular tenant, then this example directs the SQLDUMP utility to dump the data for the regTenantUser tenant's partition:

```
sqldump -u regTenantUser@OpenEdgeA -a regTenant -t mttab1 progress:T:localhost:9999:mtdb
```

If dbaUser is mapped to a DBA, then this example directs the SQLDUMP utility to dump the tenant-specific data for all the tenants in their respective directory:

```
sqldump -u dbaUser -a dba -t mttab1 progress:T:localhost:9999:mtdb
```

If superten is mapped to a super-tenant, then this example directs the SQLDUMP utility to dump all the tenants which start with the word ‘ten' for the table mttab:

```
sqldump -u superTenUser@superdom -a superten -t mttab1 -n ten%
progress:T:localhost:9999:mtdb
```

Notes

- The *db_name* must be the last parameter given.
- Before you can run SQLDUMP against a database server, the server must be configured to accept SQL connections and must be running.
- Each dump file records character set information in the identifier section of each file. For example:

```
A^B^CProgress     sqlschema       v1.0       Quote fmt
A^B^CTimestamp    1999-10-19   19:06:49:0000
A^B^CDatabase     dumpdb.db
A^B^CProgress Character Set: iso8859-1
A^B^CJava Charcter Set: Unicode UTF-8
A^B^CDate Format: MM/DD/YYYY
```

The character set recorded in the dump file is the client character set. The default character set for all non-JDBC clients is taken from the local operating system through the operating system apis. JDBC clients use the Unicode UTF-8 character set.

To use a character set different than that used by the operating system, set the SQL_CLIENT_CHARSET environment variable to the name of the preferred character set. You can define any OpenEdge supported character set name. The name is not case sensitive.

- SQLDUMP does not support the following characters in schema names:
  - Double quote (")
  - Forward slash (/)
  - Backslash (\)
- SQLDUMP supports schema names that contain special characters such as, a blank space, a hyphen (-), or pound sign (#). These names must be used as delimited identifiers. Therefore, when specifying names with special characters on a UNIX command line, follow these rules:
  - Use double quotes to delimit identifiers.
  - So that the command line does not strip the quotes, use a backslash (\) to escape the double quotes used for delimited identifiers.
  - Use double quotes to enclose any names with embedded spaces, commas, or characters special to a command shell (such as the Bourne shell). This use of quotes is in addition to quoting delimited identifiers.

  For example, to dump the table `Yearly Profits`, use the following UNIX command-line:

```
sqldump -t "\"Yearly Profits\"" -u xxx -a yyydb_name
```

- The SQLDUMP utility does not support dumping a table containing a CLOB or BLOB column. A request to dump a table with a CLOB or BLOB column will result in an error message, and that table will be skipped. To dump a table containing a CLOB or BLOB column, use the DB binary dump utility.
- In Windows, the command interpreter rules for the use of double quotation marks varies from UNIX.
- By default, SQLDUMP displays `promsgs` messages using the code page corresponding to *code_page_name*. That is, if you are dumping a Russian database, and *code_page_name* specifies the name of a Russian code page, the client displays `promsgs` messages using the Russian code-page, (unless you specify a different code page by setting the client's SQL_CLIENT_CHARSET_PROMSGS environment variable).

SQLLOAD utility

A command-line utility that loads user data from a formatted file into an SQL database.

Syntax

```
sqlload -u user_name [ -a password] [ -C code_page_name]
      -t [owner_name.] table_name1 [ [,owner_name.] table_name2, ...]
       [ -l log_file_name] [ -b badfile_name] [ -e max_errors]
       [ -F comma | quote ] [ -c commit_frequency]
       [ -n tenant_name1,...]
       db_name
```

Parameters

`u user_name`

Specifies the user SQLLOAD uses to connect to the database. If you omit the *user_name* and *password*, SQLLOAD prompts you for these parameter values. If you omit the *user_name* and supply a password, SQLLOAD uses the value defined in the USER environment variable.

`-a password`

Specifies the *password* used by the database for authentication.

`-C code_page_name`

A case-insensitive character string that specifies the name of the dump file's code page. If the `-C` parameter specifies a code page name that is not valid, a run-time error is reported. If the `-C` parameter does not appear at all, the code page name defaults to the client's internal code page:

- If set, the value of the client's SQL_CLIENT_CHARSET environment variable
- If not set, the name of the code page of the client's locale

For example, you might use the `-C` parameter to load a dump file whose code page is ISO8859-2, using a Windows client whose code page is MS1250. Although you can accomplish this by setting the client's SQL_CLIENT_CHARSET environment variable, using the `-C` parameter might be easier.

`-t owner_name.table_name`

Specifies a list of one or more tables to load into a database. This parameter is required. Pattern matching is supported, using a percent sign (%) for multiple characters and an underscore (_) for a single character. The pattern matching follows the standard for the `LIKE` predicate in SQL. You can load a single table, a set of tables, or all tables. If you omit the optional *owner_name* table qualifier, SQLLOAD uses the name specified by the `-u` parameter. The files from which SQLLOAD loads data are not specified in the SQLLOAD syntax. The utility requires that the filename follow the naming convention *owner_name*.*table_name*.dsql.

`-l log_file_name`

Specifies the file to which SQLLOAD writes errors and statistics. The default is standard output.

`-b badfile_name`

Specifies the file where SQLLOAD writes rows that were not loaded.

`-e max_errors`

Specifies the maximum number of errors that SQLLOAD allows before term processing. The default is 50.

`-F comma | quote`

Directs SQLLOAD to load data in comma-delimited format or quote-delimited format. The default is quote.

`-c commit_frequency`

Specifies the number of records written to the database before a commit is performed. Committed records cannot be rolled back if an error occurs during the load.

`-n tenant_name`

Specifies a list of one or more tenants to dump to a file. This parameter is optional.

db_name

Identifies the database where you are loading tables. You can load tables into a single database each time you invoke SQLLOAD. There is no option flag preceding the *db_name*. This parameter is required, and must be the last parameter specified. The database name is specified in the following way: `progress:T:`*localhost*`:`*demosv*`:`*jo*`.`

SQLLOAD loads user data from a formatted file into an SQL database. Typically, the source file for the load is created by executing the SQLDUMP utility. The SQLLOAD utility can process a source file created by another application or utility, if the format of the file conforms to SQLLOAD requirements. The file extension made available to SQLLOAD for processing must be `.dsql`. See the entry on SQLDUMP for a description of the required file format.

The SQLLOAD utility reads application data from variable-length text-formatted files and writes the data into the specified database. The column order is identical to the table column order. SQLLOAD reads format and content header records from the dump file. You can load multiple tables in a single execution by specifying multiple table names, separated by commas. Data for one table is from a single dump file. Every source file corresponds to one database table. For example, if you specify 200 tables in the SQLLOAD command, you will load 200 database tables.

The format of the records in the input files is similar to the ABL `.d` file dump format. See SQLDUMP utility on page 849 for a description of the record format. The maximum record length SQLLOAD can process is 32K.

Each database record read is share-locked for consistency. You must ensure that the SQL Server has a lock table large enough to contain one lock for every record in the table. The default lock table size is 10,000 locks.

SQLLOAD writes any errors to standard output and halts the loading process for any error so that data integrity is not compromised.

Example

This example directs the SQLLOAD utility to load the data from two dump files into the `salesdb` database. The input files to SQLLOAD must be `tucker.customers.dsql` and `tucker.products.dsql`, as shown:

```
sqlload -u tucker -a sulky -t tucker.customers,tucker.products
progress:T:thunder:4077:salesdb
```

This example directs SQLLOAD to load the data from all appropriately named dump files into the specified tables in the salesdb database:

```
sqlload -u tucker -a sulky -t %.cust%,%.invent%,%.sales%
progress:T:thunder:4077:salesdb
```

This example illustrates the creation of tenant directories when the tenants are dumped to a formatted file using SQLLOADutility:

```
sqlload -u supertenUser@superdom -a superten  -t mttab1 -n ten1, ten2
progress:T:localhost:9999:mtdb
```

If the `regTenant` is mapped to a regular tenant, then this example directs the SQLLOAD utility to load the data of the partition of the tenant to `mttab1` table:

```
sqlload -u regTenantUser@OpenEdgeA -a regTenant -t mttab1
progress:T:localhost:9999:mtdb
```

If the dbaUser is a DBA, then this example directs the SQLLOAD utility to load the tenant-specific data for all the tenants in their respective directories:

```
sqlload -u tucker -a sulky -t %.cust%,%.invent%,%.sales%
progress:T:thunder:4077:salesdb
```

If the superTenUser is a super-tenant, then this example directs the SQLLOAD utility to load all the data associated with the tenants which start with the word ‘ten' of the table mttab1:

```
sqlload -u superTenUser@superdom -a superten -t mttab1 -n ten%
progress:T:localhost:9999:mtdb
```

If the `dbaUser` is a DBA, then this example directs the SQLLOAD utility to load all the data for tenants for the table mttab1:

```
sqlload -u superTenUser@superdom -a superten -t mttab1 -n ten%
progress:T:localhost:9999:mtdb
```

Notes

- The `db_name` must be the last parameter given.
- Before you can run SQLLOAD against a database server, the server must be configured to accept SQL connections and must be running.
- The character set used by SQLLOAD must match the character set information recorded in each dump file. If the character sets do not match, the load is rejected. You can use the SQL_CLIENT_CHARSET environment variable to specify a character set.

Each dump file you create with SQLDUMP contains character set information about that file. The character set recorded in the dump file is the client character set. The default character set for all non-JDBC clients is taken from the local operating system through the operating system APIs. JDBC clients use the Unicode UTF-8 character set.

To use a character set different than that used by the operating system, set the SQL_CLIENT_CHARSET environment variable to the name of the preferred character set. You can define any OpenEdge-supported character set name. The name is not case sensitive.

- At run time, SQLLOAD reports an error if it detects a mismatch between the code page of the dump file being loaded and the code page of the client running SQLLOAD.
- By default, SQLLOAD displays `promsgs` messages using the code page corresponding to `code_page_name`. That is, if you are restoring a Russian database and *`code_page_name`* specifies the name of a Russian code page, the client displays `promsgs` messages using the Russian code-page (unless you specify a different code page by setting the client's SQL_CLIENT_CHARSET_PROMSGS environment variable).
- SQLLOAD does not support the following characters in schema names:
  - Double quote (`"`)
  - Forward slash (`/`)
  - Backslash (`\`)
- SQLLOAD supports schema names that contain special characters, such as a blank space, a hyphen (-), or pound sign (`#`). These names must be used as delimited identifiers. Therefore, when specifying names with special characters on a UNIX command line, follow these rules:
  - Use double quotes to delimit identifiers.
  - Use a backslash (`\`) to escape the double quotes used for delimited identifiers.
  - Use double quotes to enclose any names with embedded spaces, commas, or characters special to a command shell (such as the Bourne shell). This use of quotes is in addition to quoting delimited identifiers.

  For example, to load the table `Yearly Profits`, use the following UNIX command-line:

```
sqlload -u xxx -a yyy -t "\"Yearly Profits\"" db_name
```

- In Windows, the command interpreter rules for the use of double quotation marks varies from UNIX.

SQLSCHEMA utility

A command-line utility that writes SQL database schema components to an output file selectively.

Syntax

```
sqlschema -u user_name [ -a password]
          [ -t [owner_name.] table_name1 [,owner_name.] table_name2, . . . ]
          [ -p [owner_name.] procedure_name, . . . ]
          [ -T [owner_name.] trigger_name, . . . ]
          [ -G [owner_name.] procedure_name, . . . ]
          [ -g [owner_name.] table_name, . . . ]
          [ -s [owner_name.] tablename, . . . ]
          [ -o output_file_name]
          [ -d domain_name]
          [ -n tenant_name]
          [ -q sequence_name]
          db_name
```

Parameters

-u *user_name*

Specifies the user id that SQLSCHEMA employs to connect to the database. If you omit the *user_name* and *password*, SQLSCHEMA prompts you for these values. If you omit the *user_name* and supply a password, SQLSCHEMA uses the value defined by the USER environment variable.

-a *password*

Specifies the *password* used by the database for authentication.

-t *owner_name.table_name*

A list of one or more tables you want to capture definitions for. Pattern matching is supported, using a percent sign (%) for multiple characters and an underscore (_) for a single character. The pattern matching follows the standard for the LIKE predicate in SQL. You can write the definition for a single table, a set of tables, or all tables. If you omit the optional *owner_name* table qualifier, SQLSCHEMA uses the name specified by the -u parameter.

-p *owner_name.procedure_name*

A list of one or more procedures you want to capture definitions for. The SQLSCHEMA utility supports pattern matching for multiple and single characters. See the *owner_name.table_name* parameter for an explanation of pattern matching. You can capture the definitions for a single procedure, a set of procedures, or all procedures. If you omit the optional *owner_name* table qualifier, SQLSCHEMA uses the name specified by the -u parameter.

-T *owner_name.trigger_name*

A list of one or more triggers you want to capture definitions for. The SQLSCHEMA utility supports pattern matching for multiple and single characters. See the *owner_name.table_name* parameter for an explanation of pattern matching. You can capture the definition for a single trigger, a set of triggers, or all triggers. If you omit the optional *owner_name* table qualifier, SQLSCHEMA uses the name specified by the -u parameter.

`-G` *owner_name.procedure_name*

Allows you to dump privileges on stored procedures in the form of `GRANT` statements.

`-g` *owner_name.table_name*

A list of one or more tables whose related privileges are captured as grant statements. You can write grant statements for both column and table privileges. The utility supports pattern matching for this parameter.

`-s` *owner_name.table_name*

Specifies a list of one or more tables whose related synonyms are captured as create synonym statements. The utility supports pattern matching for this parameter.

`-o` *output_file_name*`.dfsql`

Specifies the output file where SQLSCHEMA writes the definitions. When specified, the file extension name must be `.dfsql`. If *output_file_name* is omitted, SQLSCHEMA writes the definitions to the screen.

`-d` *domain_name*

Specifies the domain name of a multi-tenant database where SQLSCHEMA writes the definitions.

`-n` *tenant_name*

Specifies the tenant name of a multi-tenant database where SQLSCHEMA writes the definitions.

`-q` *sequence_name*

Specifies the sequence name of a multi-tenant database where SQLSCHEMA writes the definitions.

db_name

Identifies the database from which SQLSCHEMA captures component definitions. You can process a single database each time you invoke SQLSCHEMA. There is no option flag preceding the *db_name*. This parameter is required and must be the last parameter specified. The database name is specified in a connection string, such as `progress:T:`*localhost:demosv:jo*`.`

SQLSCHEMA writes SQL database schema components to an output file selectively. You can capture table definitions including table constraints, views, stored procedures including related privileges, and triggers. At the command line you specify which components to dump. To load database schema information into a database, use the SQL Explorer tool. See *OpenEdge Data Management: SQL Reference* for information about SQL Explorer.

The SQLSCHEMA utility cannot write definitions for ABL tables. Table definitions include the database area name for the table, derived from a scan of the area and objects. When SQLSCHEMA writes a table definition, it does not automatically write associated triggers, synonyms, or privileges. These must be explicitly specified on the command line. Capturing database schema requires privileges to access the requested components.

Example

This example directs the SQLSCHEMA utility to write table definitions and trigger information. The output goes to the screen since no *output_file_name* is specified. Since the user name and password are not specified, SQLSCHEMA will prompt the user for these values, as shown:

```
sqlschema -t tucker.customers,tucker.products -T
tucker.customers,tucker.products progress:T:thunder:4077:salesdb
```

This example directs the SQLSCHEMA utility to write table definitions to an output file named `salesdbschema.dfsql`:

```
sqlschema -u tucker -a sulky -t %.cust%,%.invent%,%.sales% -o
salesdbschema.dfsql progress:T:thunder:4077:salesdb
```

This example directs the SQLSCHEMA utility to write schema definitions for two domains `OpenEdgeA` and `OpenEdgeB` respectively:

```
sqlschema -u mtuser@mtdomain -a mtuser -d OpenEdgeA, OpenEdgeB
progress:T:localhost:2222:sportsdb
```

This example directs the SQLSCHEMA utility to write schema definitions for domains with `Access` as its last 6 characters:

```
sqlschema -u mtuser@mtdomain -a mtuser -d  %Access
progress:T:localhost:2222:sportsdb
```

This example directs the SQLSCHEMA utility to write schema definitions for domains with `OpenEdge` as its first 8 characters followed by a single character:

```
sqlschema -u mtuser@mtdomain -a mtuser -d OpenEdge_
progress:T:localhost:2222:sportsdb
```

This example directs the SQLSCHEMA utility to write schema definitions for tenants `T1` and `T2` respectively:

```
sqlschema -u mtuser@mtdomain -a mtuser -n T1, T2
progress:T:localhost:2222:sportsdb
```

This example directs the SQLSCHEMA utility to write schema definitions for the two sequences `seq1` and `seq2`:

```
sqlschema -u mtuser@mtdomain -a mtuser -Q seq1, seq2
progress:T:localhost:2222:sportsdb
```

This example directs the SQLSCHEMA utility to write schema definitions for sequences that start with `seq` followed by a single character:

```
sqlschema -u mtuser@mtdomain -a mtuser -Q seq_
progress:T:localhost:2222:sportsdb
```

Notes

- Before you can run SQLSCHEMA against a database server, the server must be configured to accept SQL connections and must be running.
- Each output file created by the SQLSCHEMA utility records character set information about the contents of the file. When you use SQLSCHEMA to dump schema information from a database, the schema is written in Unicode UTF-8.

30

Virtual System Tables

Virtual system tables (VSTs) give ABL and SQL applications access to the same type of database information that you collect with the OpenEdge Monitor (PROMON) utility. Virtual system tables enable an application to examine the status of a database and monitor its performance. With the database broker running, ABL and SQL applications can query a VST and retrieve the specified information as run-time data.

For details, see the following topics:

- Update access to virtual system tables
- Accessing VSTs with ABL or SQL
- Modifiable VSTs on Replication Targets
- Virtual system table summaries

Update access to virtual system tables

The OpenEdge RDBMS provides the empty, demo, and sports2000 databases with the virtual system table schemas already loaded. As new virtual system tables are made available, you can update the schemas in a database.

To update the schemas, run the following PROUTIL command before you start the database server, where *db-name* specifies the database you are using.: `proutil db-name -C updatevst`

Accessing VSTs with ABL or SQL

You can write ABL or SQL to query the information contained in the Virtual System Tables. However, because of the leading underscore (_) in the table names, and field or column names, quotes must be used when writing SQL queries. The following ABL and SQL statements both determine the enabled features of a database from the _database-Feature VST:

Table 113: ABL query

```
FOR EACH _database-Feature:
     IF _DBFeature_Enabled = "1" THEN
     DISPLAY _DBFeature_Name.
```

Table 114: SQL query

```
SELECT
"_DBFeature_name" FROM pub."_database-Feature"
     WHERE "_DBFeature_Enabled" = 1;
```

The result of running either query against the sports database is:

```
_DBFeature_Name
------------------------
64 Bit DBKEYS
Large Keys
64 Bit Sequences
```

See the sections that follow for descriptions of the Virtual System Tables and their fields.

Modifiable VSTs on Replication Targets

To ease Virtual System Table (VST) updating on replication targets, the following modifiable VST tables and fields are allowed on replication targets.

Database connection (_Connect)

Field name	Data type	Description
_Connect-Misc	CHARACTER	User updatable field.

User connection (_MyConnection)

Table 115:

Field name	Data type	Description
_MyConn-UserMisc	CHARACTER	User updatable field.

Database features (_Database-Feature)

Field Name	Data Type	Description
_DBFeature_Enabled	CHARACTER	"1" indicates the feature is enabled; "0" indicates the feature is not enabled

Note: The only feature that may be enabled on a replication target is Large Database Entry Keys. It may only be enabled.

Database startup parameters file status (_DbParams)

Field name	Data type	Description
_DbParams-Value	CHARACTER	The current value specified for the parameter

Note: The following list contains the Database Startup Parameters that may have their value fields modified on a replication target. Other Startup Parameters are fixed and may not be modified.

- Base Index (-baseindex)
- Base LOB (-baselob)
- Base Table (-basetable)
- Base User Index (-baseuserindex)
- Base User LOB (-baseuserlob)
- Base User Table (-baseusertable)
- BI Writer Scan Delay (-bwdelay)
- Diagnostic Directory (-diagDir)
- Diagnostic Events Value (-diagEvent)
- Diagnostic Events Level (-diagEvtLevel)
- Diagnostic Field Separator (-diagFS)
- Diagnostic Data Format (-diagFormat)
- Diagnostic Pause Length (-diagPause)
- Diagnostic Prefix Value (-diagPrefix)
- Log File Archive Directory (-lgArchiveDir)
- Log File Archive Enable (-lgArchiveEnable)
- Log File Truncate Frequency (-lgTruncateFrequency)
- Log File Truncate Size (-lgTruncateSize)
- Log File Truncate Time (-lgTruncateTime)
- Limit Log File Payload (-limitLgPayload)
- LRU Force Skips (-lruskips)
- LRU Alternate Buffer Pool Force Skips (-lru2skips)
- Nap Time (-nap)
- Nap Maximum (-napmax)
- Login Governor (-nGovernor)
- Number of Checkpoint Statistics (-numcheckpointstats)
- Omit Log Messages (-omitLgMsgs)
- PICA Queue Nap Time (-picanap)
- Prefetch Delay (-prefetchDelay)
- Prefetch Factor (-prefetchFactor)
- Prefetch Num Recs (-prefetchNumRecs)
- Prefetch Priority (-prefetchPriority)
- Page Writer Queue Delay (-pwqdelay)

- Page Writer Queue Minimum (`-pwqmin`)
- Page Writer Scan Delay (`-pwsdelay`)
- Page Writer Scan (`-pwscan`)
- Page Writer Maximum Buffers (`-pwwmax`)
- Record Free Chain Search Depth Factor (`-recspacesearchdepth`)
- Spin Lock Retries (`-spin`)
- Server-side JOIN (`-ssj`)
- Database Client Notification (`-usernotifytime`)
- Database Consistency Check (`-DbCheck`)
- Lock Governor (`-LGovernor`)
- Memory Consistency Check (`-MemCheck`)
- Message Wait (`-Nmsgwait`)
- Autonomous Update Statistics (`-SQLAutoStats`)
- Transaction End (TXE) Lock Retry Limit (`-TXERetryLimit`)
- Transaction End (TXE) Commit Lock Skip Limit (`-TXESkipLimit`)

Virtual system table summaries

This sections that follow, are an alphabetical summary of the OpenEdge RDBMS's virtual system tables.

After-image log activity (_ActAILog)

The _ActAILog VST displays after-image log activity, such as the number of after-image writes, records and bytes written, busy buffer waits, and log force waits.

Field name	Data type	Description
_AiLog-AIWWrites	INT64	Number of after-image writes performed by the after-image writer (AIW). This is a subset of the total number of AI writes.
_AiLog-BBuffWaits	INT64	Number of busy buffer waits.
_AiLog-BytesWritn	INT64	Amount of AI data written to the AI file, in bytes.
_AiLog-ForceWaits	INT64	Number of waiting-for-commit records to be written to disk.
_AiLog-NoBufAvail	INT64	Total number of times a process had to wait because a buffer was not available.
_AiLog-PartialWrt	INT64	Number of writes to the AI file made before the AI buffer is full.

Field name	Data type	Description
_AiLog-RecWriten	INT64	Number of records written to the AIfile.
_AiLog-TotWrites	INT64	Total number of writes to the AI file.
_AiLog-Trans	INT64	Number of transactions committed to the AI file.
_AiLog-UpTime	INTEGER	Number of seconds the AI file was open.

Before-image log activity (_ActBILog)

The _ActBILog VST displays before-image log activity, such as the number of before-image reads and writes, records and bytes written, number of records and bytes read and the number of busy and empty buffer and log force waits.

Field name	Data type	Description
_BiLog-BOBufHit	INT64	Number of input buffer hits
_BiLog-BBuffWaits	INT64	Number of times a process had to wait for a buffer that was busy.
_BiLog-BIWWrites	INT64	Number of writes to the BI file performed by the before-image writer (BIW). For good performance, this number should be high in relation to the total number of BI writes.
_BiLog-BytesRead	INT64	Number of bytes of data read from the BI file.
_BiLog-BytesWrtn	INT64	Number of bytes of data written to the BI file.
_BiLog-ClstrClose	INT64	Number of BI clusters filled and closed in preparation for reuse.
_BiLog-EBuffWaits	INT64	Number of times a process had to wait because all buffers were full.
_BiLog-ForceWaits	INT64	Number of waiting-for-commit records to be written to disk.
_BiLog-ForceWrts	INT64	Number of waiting-for-commit records written to disk.
_BiLog-InputBufHit	INT64	Number of input buffer hits
_BiLog-ModBufHit	INT64	Number of modified buffer hits
_BiLog-OutputBufHit	INT64	Number of output buffer hits
_BiLog-PartialWrts	INT64	Number of writes to the BI file made before the BI buffer is full.

Field name	Data type	Description
_BiLog-RecRead	INT64	Number of BI records (notes) read from the BI file.
_BiLog-RecWriten	INT64	Number of BI records (notes) written to the BI file.
_BiLog-TotalWrts	INT64	Number of total writes to the BI file.
_BiLog-TotReads	INT64	Number of BI blocks read from the BI file to undo transactions.
_BiLog-Trans	INT64	Number of transactions committed to the BIfile.
_BiLog-UpTime	INTEGER	Number of seconds the BI file was open.

Buffer activity (_ActBuffer)

The _ActBuffer VST displays the activity of the database buffer cache, such as the number of logical reads and writes, OS reads and writes, checkpoints, deferred writes, LRU skips and writes, and APW enqueues.[34]

Note: The _ActBuffer table, which contains buffer activity data, holds three records: the first record contains the totals of both the Alternate and primary buffer pools; the second record contains data on the primary buffer pool; the third record contains data on the Alternate Buffer Pool

Field name	Data type	Description
_Buffer-Active	INT64	Number of buffers active in each buffer pool
_Buffer-APWEnq	INT64	Number of modified buffers placed on the APW queue for writing.
_Buffer-Area	INT64	Area number
_Buffer-BlockMap	INT64	Block allocation bit map block (Contains a bit map of block allocation within the object. It is allocated, but is currently not used. Accesses can show up in the buffer pool.)
_Buffer-Chkpts	INT64	Number of checkpoint operations.
_Buffer-ClusterMap	INT64	Cluster bit Map Blocks (Contains a bit map of cluster allocation within the area. It is allocated, but is currently not used. Accesses can show up in the buffer pool.)
_Buffer-Control	INT64	Number of control blocks.

[34] Activity and status of ABL temp-tables is available in this virtual system table. The statistics gathered allow you to monitor the operation and performance of deployed client temp-tables. Specify the range of temp-tables to track with the `-ttbasetable` and `-tttablerangesize` startup parameters. Specify temp table indexes with `-ttbaseindex` and `-ttindexrangesize` startup parameters. For more information monitoring and debugging temp tables with VSTs, see *OpenEdge Development: Debugging and Troubleshooting*.

Field name	Data type	Description
_Buffer-Deferred	INT64	Total number of changes to blocks that occurred before the blocks were written. Each deferred write is potentially an I/O operation saved.
_Buffer-Free	INT64	Number of free buffers
_Buffer-Flushed	INT64	Number of blocks that were not written during the checkpoint and that had to be written all at once at the end of the checkpoint.
_Buffer-Index	INT64	Number of index blocks buffered
_Buffer-LogicRds	INT64	Number of client requests for database block read operations.
_Buffer-LogicWrts	INT64	Number of client requests for database block write operations.
_Buffer-LRUEnabled	LOGICAL	True when all the buffers in the Alternate Buffer Pool (`-B2`) are consumed and an LRU replacement algorithm is utilized for buffer eviction.
_Buffer-LRUSkips	INT64	Number of times a buffer on the LRU chain was skipped because it was locked or modified.
_Buffer-LRUwrts	INT64	Number of blocks written to free a buffer for a read operation.
_Buffer-Marked	INT64	Number of blocks scheduled to be written before the end of a checkpoint.
_Buffer-Master	INT64	Buffer identifier. Valid values are: • 0 — Primary buffer pool • 1 — Secondary buffer pool
_Buffer-Object	INT64	Number of object blocks
_Buffer-ObjClist	INT64	Object Cluster List Blocks (contains the list of clusters owned by a particular object)
_Buffer-ObjList	INT64	Number of object list blocks
_Buffer-OSRds	INT64	Number of database blocks read from disk
_Buffer-OSWrts	INT64	Number of database block writes to disk
_Buffer-RM	INT64	Number of record blocks buffered
_Buffer-Seq	INT64	Number of sequence blocks

Field name	Data type	Description
_Buffer-Trans	INT64	Number of transactions committed
_Buffer-Uptime	INTEGER	Number of seconds the database was opened

Index activity (_ActIndex)

The _ActIndex VST displays index activity, such as the number of entry finds, creates, and deletes, the number of locked entries removed, and the numbers of split and free blocks.[34]

Field name	Data type	Description
_Index-Create	INT64	Number of new index entries generated
_Index-Delete	INT64	Number of index entries deleted
_Index-Find	INT64	Number of times an index entry was looked up
_Index-Free	INT64	Number of index blocks freed during deletes
_Index-PartitionId	INTEGER	Partition Id of the index
_Index-Remove	INT64	Number of locks released at transaction end
_Index-Splits	INT64	Number of block splits resulting from index additions
_Index-Trans	INT64	Number of transactions committed
_Index-UpTime	INTEGER	Number of seconds the database was up

Input/output activity (_ActIOFile)

The _ActIOFile VST displays information about input/output activity, including the number of reads, writes, and extends for each file.[34]

Field name	Data type	Description
_IOFile-BufReads	INT64	Number of buffered read operations performed on the file
_IOFile-BufWrites	INT64	Number of buffered write operations performed on the file
_IOFile-Extends	INT64	Number of extend operations performed on the file
_IOFile-FileName	CHARACTER	Name of the file upon which operations are performed
_IOFile-Reads	INT64	Number of read operations performed on the file

Field name	Data type	Description
_IOFile-Trans	INT64	Number of transactions committed
_IOFile-UbufReads	INT64	Number of unbuffered read operations performed on the file
_IOFile-UbufWrites	INT64	Number of unbuffered write operations performed on the file
_IOFile-UpTime	INTEGER	Number of seconds the database was up
_IOFile-Writes	INT64	Number of write operations performed on the file

Input/output type activity (_ActIOType)

The _ActIOType VST displays information about types of input/output activity, such as database reads and writes, before-image and after-image reads, total reads, before-image and after-image writes, committed transactions, and database up time.[34]

Field name	Data type	Description
_IOType-AiRds	INT64	Number of AI read operations
_IOType-AiWrts	INT64	Number of AI write operations
_IOType-BiRds	INT64	Number of BI read operations
_IOType-BiWrts	INT64	Number of BI write operations
_IOType-DataReads	INT64	Number of reads of data blocks
_IOType-DataWrts	INT64	Number of writes to data blocks
_IOType-IdxRds	INT64	Number of reads of index blocks
_IOType-IdxWrts	INT64	Number of writes to index blocks
_IOType-Trans	INT64	Number of transactions committed
_IOType-UpTime	INTEGER	Number of seconds the database was up

Lock table activity (_ActLock)

The _ActLock VST displays lock table activity, including the number of share, exclusive, upgrade, Rec Get, and redundant requests; the number of exclusive, Rec Get, share, and upgrade grants; the number of exclusive, Rec Get, share, and upgrade waits; the number of downgrades, transactions committed, cancelled requests, and database up time.

Field name	Data type	Description
_Lock-CanclReq	INTEGER	Number of lock requests that were cancelled
_Lock-Downgrade	INTEGER	Number of locks that were downgraded
_Lock-ExclFind	INT64	Number of exclusive find requests granted
_Lock-ExclLock	INTEGER	Number of exclusive lock requests
_Lock-ExclReq	INTEGER	Number of requests for exclusive locks granted
_Lock-ExclWait	INTEGER	Number of times processes waited for an exclusive lock
_Lock-RecGetLock	INTEGER	Number of requests for record get locks granted
_Lock-RecGetReq	INTEGER	Number of record get locks granted
_Lock-RecGetWait	INTEGER	Number of times processes waited for record gets
_Lock-RedReq	INTEGER	Number of duplicate requests granted
_Lock-ShrFind	INT64	Number of share find requests granted
_Lock-ShrLock	INTEGER	Number of share lock requests granted
_Lock-ShrReq	INTEGER	Number of requests for share locks
_Lock-ShrWait	INTEGER	Number of times processes waited for share lock
_Lock-Trans	INT64	Number of transactions committed
_Lock-UpgLock	INTEGER	Number of requests for lock upgrades granted
_Lock-UpgReq	INTEGER	Number of requests to upgrade a lock from shared lock to exclusive lock
_Lock-UpgWait	INTEGER	Number of times processes waited for upgrades
_Lock-UpTime	INTEGER	Number of seconds the database was up

Other activity (_ActOther)

The _ActOther VST displays information about miscellaneous activity, including the number of commits, undo operations, semaphore waits, master block flushes, and database up time.[34]

Field name	Data type	Description
_Other-Commit	INT64	Number of transactions committed

Field name	Data type	Description
_Other-FlushMblk	INT64	Number of times the database master block was written to disk
_Other-NBWait	INT64	Number of non-blocking waits for remote clients
_Other-SemLatWait	INT64	Number of semaphore latch waits
_Other-Trans	INT64	Transactions committed
_Other-Undo	INT64	Number of transactions rolled back
_Other-UpTime	INTEGER	Number of seconds the database was up
_Other-Wait	INT64	Number of times a process had to wait for a resource

Page writer activity (_ActPWs)

The _ActPWs VST displays information about asynchronous page writer (APW) activity, including the number of APW queue and database writes, checkpoint and scan writes, total database writes; the number of buffers scanned and checkpointed; the number of checkpoints, checkpoint flushes, and marks; the number of scan cycles, committed transactions, and database up time.

Field name	Data type	Description
_PW-ApwQWrites	INT64	Number of buffers written to clear the APW queue
_PW-BuffsScaned	INT64	Number of buffers scanned during each cycle
_PW-BufsCkp	INT64	Number of buffers checkpointed
_PW-Checkpoints	INT64	Number of checkpoints that have occurred
_PW-CkpQWrites	INT64	Number of buffers written from the checkpoint queue
_PW-DBWrites	INT64	Number of database write operations performed by the APW
_PW-Flushed	INT64	Number of blocks that were not written during the checkpoint and had to be written all at once at the end of the checkpoint
_PW-Marked	INT64	Number of buffers that were scheduled to be written before the end of the checkpoint
_PW-ScanCycles	INT64	Number of scan cycles. During a scan cycle, the APW scans a portion of the buffer pool to look for modified buffers
_PW-ScanWrites	INT64	Number of buffers written during the scan cycle

Field name	Data type	Description
_PW-TotDBWrites	INT64	Total number of database write operations performed by all processes
_PW-Trans	INT64	Number of transactions committed
_PW-UpTime	INTEGER	Number of seconds the database was up

Record activity (_ActRecord)

The _ActRecord VST displays record activity information, including the number of bytes created, deleted, read, and updated; the number of fragments created, deleted, read, and updated; the number of records created, deleted, read, and updated; the number of transactions committed; and database up time.[34]

Field name	Data type	Description
_Record-BytesCreat	INT64	Number of bytes created
_Record-BytesDel	INT64	Number of bytes deleted
_Record-BytesRead	INT64	Number of bytes read
_Record-BytesUpd	INT64	Number of bytes updated
_Record-FragCreat	INT64	Number of fragments created
_Record-FragDel	INT64	Number of fragments deleted
_Record-FragRead	INT64	Number of fragments read
_Record-FragUpd	INT64	Number of fragments updated
_Record-RecCreat	INT64	Number of records created
_Record-RecDel	INT64	Number of records deleted
_Record-RecRead	INT64	Number of records read
_Record-RecUpd	INT64	Number of records updated
_Record-Trans	INT64	Number of transactions committed
_Record-UpTime	INTEGER	Number of seconds the database was up

Server activity (_ActServer)

The _ActServer VST displays server activity information, including the number of bytes sent and received, the number of messages sent and received, the number of queries received, the number of records sent and received, the number of query time slice switches, the number of transactions committed, and database up time.[34]

Field name	Data type	Description
_Server-ByteRec	INT64	Number of bytes received by the server
_Server-Id	INT64	Server ID
_Server-ByteSent	INT64	Number of bytes sent by the server
_Server-MsgRec	INT64	Number of network packets received by the server
_Server-MsgSent	INT64	Number of network packets sent by the server
_Server-QryRec	INT64	Number of query requests received
_Server-RecRec	INT64	Number of records received
_Server-RecSent	INT64	Number of records sent
_Server-TimeSlice	INT64	Number of query time slice switches
_Server-Trans	INT64	Number of transactions committed
_Server-UpTime	INTEGER	Number of seconds the database was up

Space allocation activity (_ActSpace)

The _ActSpace VST displays space allocation information, including the number of database extends, the number of times a block was used from and returned to the free chain, the number of times space was allocated for a record (from the rm chain or from the free chain) the number of bytes allocated for record fragments, the number of rm blocks examined or removed, the number of blocks added to the front or back of the rm chain, the number of moved blocks, the number of locked chain entries, the number of transactions committed, and database up time.[34]

Field name	Data type	Description
_Space-AllocNewRm	INT64	Number of times space was allocated for a record or record fragment
_Space-BackAdd	INT64	Number of blocks added to the back of the rm chain
_Space-BytesAlloc	INT64	Number of bytes allocated for record fragments
_Space-DbExd	INT64	Number of times the database was extended
_Space-Examined	INT64	Number of blocks examined in the rm chain while looking for space for a record fragment
_Space-FromFree	INT64	Number of times space was allocated from the free chain
_Space-FromRm	INT64	Number of times space was allocated from the rm chain

Field name	Data type	Description
_Space-Front2Back	INT64	Number of blocks moved from the front to the back of the rm chain
_Space-FrontAdd	INT64	Number of blocks added to the front of the rm chain
_Space-Locked	INT64	Number of locked rm entries removed
_Space-Removed	INT64	Number of blocks removed from the rm chain
_Space-RetFree	INT64	Number of times a block was returned to the free chain
_Space-TakeFree	INT64	Number of times a block was used from the free chain
_Space-Trans	INT64	Number of transactions committed
_Space-UpTime	INTEGER	Number of seconds the database was up

Summary activity (_ActSummary)

The _ActSummary VST displays general information about database activity, including the number of transactions committed and rolled back; the number of records read, updated, created, and deleted; the number of record locks and waits; the number of database reads and writes; before-image and after-image information; and buffer information.[34]

Field name	Data type	Description
_Summary-AiWrites	INT64	Number of AI blocks written to disk
_Summary-BiReads	INT64	Number of BI blocks read
_Summary-BiWrites	INT64	Number of BI blocks written to disk
_Summary-Chkpts	INT64	Number of checkpoints that have been performed
_Summary-Commits	INT64	Number of transactions all users have committed
_Summary-DbAccesses	INT64	Number of times users have waited for shared and exclusive locks on a database buffer
_Summary-DbReads	INT64	Number of database blocks read
_Summary-DbWrites	INT64	Number of database blocks written to disk
_Summary-Flushed	INT64	Number of database buffers that have been flushed to disk because they were not written by the time the checkpoint ended
_Summary-RecCreat	INT64	Number of records created

Field name	Data type	Description
_Summary-RecDel	INT64	Number of records deleted
_Summary-RecLock	INT64	Number of record locks used
_Summary-RecReads	INT64	Number of records read
_Summary-RecUpd	INT64	Number of records updated
_Summary-RecWait	INT64	Number of times users have waited to access a locked record
_Summary-TransComm	INT64	Number of transactions committed
_Summary-Undos	INT64	Number of transactions rolled back
_Summary-Uptime	INTEGER	Number of seconds the database was up

Area status (_AreaStatus)

The _AreaStatus VST displays a variety of data about the status of areas.

Field name	Data Type	Description
_AreaStatus-AIActivated	CHARACTER	The the date/time that AI was activated
_AreaStatus-AI-Seq	INTEGER	The sequence number of each AI file
_AreaStatus-State	CHARACTER	The state of each AI file in the database. Valid values are: • EMPTY • FULL • BUSY • LOCKED
_AreaStatus-Areaname	CHARACTER	Area name
_AreaStatus-Areanum	INTEGER	Area number
_AreaStatus-Extents	INTEGER	Number of extents in an area
_AreaStatus-Freenum	INT64	Number of blocks on the free chain
_AreaStatus-Hiwater	INT64	Highest block used
_AreaStatus-Lastextent	CHARACTER	Name of extent where high-water mark resides
_AreaStatus-LastOp	CHARACTER	The date/time of the last operation on this area

Field name	Data Type	Description
_AreaStatus-Rmnum	INT64	Number of blocks on the rm chain
_AreaStatus-Totblocks	INT64	Total number of blocks in an area
_AreaStatus-Type	CHARACTER	The type of area. Valid values include: • AI • BI • TL • Data • Control

Area threshold (_AreaThreshold)

The _AreaThreshold VST displays information about the consumption of available address space in an area.

Field name	Data type	Description
_AreaThreshold	INTEGER	Number indicating how much of the addressable space in an area is consumed; possible values are: • 0 — Area is below 80 percent of maximum addressable threshold value. • 1 — Area is above 80 percent of maximum addressable threshold value. • 2 — Area is above 90 percent of maximum addressable threshold value. • 4 — Area has reached the maximum addressable threshold and cannot be further extended.
_AreaThresholdArea	INTEGER	Area number

Block (_Block)

The _Block VST displays information about a specific block.[34]

Field name	Data type	Description
_Block-Area	INTEGER	Block area number
_Block-BkupCtr	INTEGER	Backup counter
_Block-Block	CHARACTER	Data section of the block
_Block-ChainType	CHARACTER	Chain type

Field name	Data type	Description
_Block-Dbkey	INT64	Dbkey
_Block-NextDbkey	INT64	Next Dbkey in the appropriate chain
_Block-Type	CHARACTER	Type of block
_Block-Update	INTEGER	Number of times the block has been updated

Buffer status (_BuffStatus)

The _BuffStatus VST displays the status of buffers, such as the number of buffers that are in the buffer cache, that are currently in use, that are empty, or that are on the LRU chain, page writer queue, or checkpoint queue.[34]

Field name	Data type	Description
_BfStatus-APWQ	INTEGER	Number of buffers on the page writer queue.
_BfStatus-CKPMarked	INT64	Number of buffers currently marked for checkpoint.
_BfStatus-CKPQ	INTEGER	Number of buffers on the checkpoint queue.
_BfStatus-HashSize	INTEGER	Size of the buffer hash table.
_BfStatus-LastCkpNum	INTEGER	Most recent checkpoint number. As checkpoints begin, they are assigned a sequential number from the start of the session; the number is also the number of checkpoints that have occurred.
_BfStatus-LRU	INTEGER	Number of buffers on the least recently used (LRU) chain for both the primary and Alternate buffer pools.
_BfStatus-LRU2	INTEGER	Number of buffers on the least recently used (LRU) chain for the Alternate buffer pool.
_BfStatus-ModBuffs	INT64	Number of dirty (modified) buffers.
_BfStatus-TotBufs	INTEGER	Number of buffers in the buffer cache.
_BfStatus-UsedBuffs	INTEGER	Number of buffers used.
_BfStatus-LRU-Skips	INTEGER	Number of times a buffer in the buffer pool is accessed before being placed on the end of the LRU chain.
_BfStatus-LRU2-Skips	INTEGER	Number of times a buffer in the alternate buffer pool is accessed before being placed on the end of the LRU chain.

Cache (_Cache)

The _Cache VST displays the status and activity of the CDC cache.

Field name	Data type	Description
_Cache-Name	CHARACTER	Name of the cache. Each supported cache has a different name.
_Cache-Size	INTEGER	Size of the cache in kbytes.
_Cache-Entries	INTEGER	Number of entries stored in the cache.
_Cache-Hash	INTEGER	Hash tables size or hash prime of the cache.
_Cache-Hits	INT64	Total number of cache hits since the last reset. (Use PROMON to reset.)
_Cache-Misses	INT64	Total number of cache misses since the last reset. (Use PROMON to reset.)

Checkpoint (_Checkpoint)

The _Checkpoint VST displays information about each checkpoint, including the checkpoint number and beginning time, the time required to complete the checkpoint, and the number of modified buffers scheduled to be written. The file also describes APW-written buffers.

Field name	Data type	Description
_Checkpoint-ApwQ	INT64	Number of buffers written by the APW queue and replaced on the least recently used (LRU) chain by APWs
_Checkpoint-Bi-Writes	INTEGER	The number of BI buffers written to disk at checkpoint end
_Checkpoint-Bi-Write-Time	DECIMAL	Time in seconds to write the BI buffers at checkpoint end
_Checkpoint-Buffers	INT64	The number of database buffers checkpointed
_Checkpoint-Cluster	INT64	The checkpoint current BI cluster number
_Checkpoint-CptQ	INT64	Number of buffers written from the checkpoint queue by the APWs
_Checkpoint-B-Write-Time	DECIMAL	Time in seconds to write the buffers from the database buffer pool at checkpoint end
_Checkpoint-Dirty	INT64	Number of modified buffers scheduled to be written

Field name	Data type	Description
_Checkpoint-Duration	DECIMAL	The time to do the housekeeping of the checkpoint. This includes the time to write the active recovery log buffers to disk, write any outstanding dirty buffers marked at the previous checkpoint, scan the database buffer pools for dirty buffers marking them as needing to be written to disk prior to the start of the next check point (putting them on the checkpoint queue) and to perform the file system synchronization (_Checkpoint-SyncTime)
_Checkpoint-Flush	INT64	Number of database buffers written at the end of the checkpoint
_Checkpoint-Len	CHARACTER	The timestamp value reporting the time the checkpoint completed. The checkpoint is not considered completed until all dirty buffers on the checkpoint queue have been written out. This is either done over time by the APWs between checkpoints or is forced at the next checkpoint (buffers flushed at checkpoint)
_Checkpoint-Number	INT64	The number of this checkpoint
_Checkpoint-Scan	INT64	Number of buffers written by the APWs during the scan cycle
_Checkpoint-SyncTime	DECIMAL	The time to perform the file system synchronization of the checkpoint. (`fdatasync()` on UNIX systems, `FlushFileBuffers()` on Windows systems)
_Checkpoint-Time	CHARACTER	Time the checkpoint began

Code Features (_Code-Feature)

The _Code-Feature VST displays the list of database features that the code supports and requires of all databases that are opened with it.

Field name	Data type	Description
_CodeFeature-ID	INT64	Feature number.
_CodeFeature_Name	CHARACTER	Name of the database feature.
_CodeFeature_Supported	CHARACTER	"1" indicates the code supports this database feature. "0" indicates the code does not support this feature.

Field name	Data type	Description
_CodeFeature_Required	CHARACTER	"1" indicates this feature is required to be active in any database this code opens. "0" indicates the database does not require this feature to be active.
_CodeFeature-Res01	CHARACTER	Reserved for future use.
_CodeFeature-Res02	INTEGER	Reserved for future use.

Database connection (_Connect)

The _Connect VST displays information about two-phase commit, batch user, and device connections.

Note: If you are connecting to a database on UNIX through a Telnet session using -S port -H localhost, a TTY session ID is appended to the field.

Field name	Data type	Description
_Connect-2phase	INTEGER	Displays a two-phase commit flag: • **1** — Indicates that 2-phase commit is enabled. • **0** — Indicates that 2-phase commit is not enabled.
_Connect-Batch	CHARACTER	Displays batch users status.
_Connect-CacheInfoType	CHARACTER	Displays the client database-request statement cache types, which include: • **ABL Program** — Current ABL program and line number. • **ABL Stack** — The ABL program stack. Includes up to 32 ABL fully qualified program names with line numbers. This field identifies the contents of _Connect-CacheInfo and _Connect-CacheLineNumber. • **SQL** — SQL statement.

Field name	Data type	Description
_Connect-CacheInfo[32]	CHARACTER	Contains tracing information based on the the type (_Connect-CacheInfoType): • **ABL Program** — Extent 1 contains the fully qualified ABL program with procedure name if applicable. • **ABL Stack** — Each extent contains a fully qualified ABL program name. Only the most current 32 program names are returned. • **SQL Statement** — Extent 1 contains a SQL statement.
_Connect-CacheLineNumber[32]	INTEGER	Contains line numbers based on the type (_Connect-CacheInfoType): • **ABL program** — Extent 1 contains the line number of the current ABL program. • **ABL Stack** — Each extent contains the line number of the fully qualified ABL program name. Only the most current 32 program names are returned. • **SQL Statement** — 0.
_Connect-CacheLastUpdate	CHARACTER	The current date and time of the client database-request statement cache.
_Connect-CachingType	INTEGER	The type of client database-request statement caching performed by the client. This field includes: • **1** — One ABL program or a single SQL statement is stored in the client database-request statement cache. • **2** — An ABL program stack or a single SQL statement is stored in the client database-request statement cache. • **3** — The client sends an ABL program stack or a single SQL statement to the client database-request statement cache.

Field name	Data type	Description
_Connect-ClientType	CHARACTER	The client type for remote and local clients connected to the database. For clients from releases earlier than 10.1B, the field does not have a value. For clients from Release 10.1B and later, the field contains one of the following values: • **ABL** — ABL client • **SQLC** — SQL client • **WTA** — Webspeed agent • **APSV** — AppServer agent • **SQFC** — SQL Federated client
_Connect-Device	CHARACTER	Device from which the user has connected to the database. **Note:** If you are connecting to a database on UNIX through a Telnet session using `-S port -H localhost`, a `TTY` session ID is appended to the field.
_Connect-Disconnect	INTEGER	Displays a disconnect flag.
_Connect-DomainId	INT64	Domain Id of the connection.
_Connect-Id	INT64	User number
_Connect-Interrupt	INTEGER	Displays an interrupt flag.
_Connect-IPAddress	CHARACTER	Displays the IP address of a remote client connection.
_Connect-LastNotifyCheck	CHARACTER	Datetime when the user last checked for notifications.
_Connect-Misc	CHARACTER	User updatable field.
_Connect-Name	CHARACTER	Displays the user name used for the connection.

Field name	Data type	Description
_Connect-Notifications	CHARACTER	Displays notification codes. Schema recache is indicated by **R**. Valid codes include: • **-:-** (no notifications) • **R:-** (recache notification marked, but not sent to user) • **R:R** (recache notification is marked, and the user is handling a recache notification) • **-:R** (no marked notifications, and the user is handling a recache notification) The general format is *marked-nofications:in-progress-notifications*
_Connect-NotifyTime	INTEGER	The current user notification polling time in seconds. A value of zero (0) indicates that notification is disabled.
_Connect-NumSeqBuffers	INTEGER	The number of private buffers specified by -Bp
_Connect-NumTrans	INT64	The number of transactions for this connection
_Connect-PartitionId	INTEGER	Displays the partition ID. This field is valid if the database is enabled for Table Partitioning, and the wait type is REC or RGET.
_Connect-Pid	INTEGER	Displays the process ID of the user session.
_Connect-Resync	INTEGER	Displays a flag that indicates the database is resyncing.
_Connect-SemId	INTEGER	Displays the semaphore ID.
_Connect-SemNum	INTEGER	Displays the semaphore number.
_Connect-Server	INTEGER	Identifies the server if the connection is remote.
_Connect-ServerPid	INTEGER	Displays the process ID of the client's server
_Connect-ServerTid	INTEGER	Displays the thread ID of the client's server
_Connect-TenantId	INTEGER	Tenant ID for the user

Field name	Data type	Description
_Connect-Time	CHARACTER	Time the user connected to the database.
_Connect-Tid	INTEGER	Displays the thread ID of the user session
_Connect-Timestamp	INTEGER	Timestamp of the user's schema cache.
_Connect-TransId	INTEGER	Displays the current transaction ID.
_Connect-Type	CHARACTER	Displays the connection type: SELF, REMC, BROK, SERV, or BTCH.
_Connect-UsedSeqBuffers	INTEGER	The number of private (-Bp) buffers in use
_Connect-UserMisc	CHARACTER(32)	User updateable field (32 byte maximum)
_Connect-Usr	INTEGER	Displays the user number.
_Connect-Wait	CHARACTER	Displays the shared memory resource the user is waiting on.
_Connect-Wait1	INT64	The ID of a resource the process is waiting on if _Connect-Wait has a value. For example, if waiting on a record lock, this field contains rowid:partition.
_Connect-Wait2	INTEGER	Displays additional info based on the value of _Connect-Wait1. If _Connect-Wait1 contains a ROWID, this field contains a table number. If _Connect-Wait1 contains a dbkey, this field contains an area number.

Database features (_Database-Feature)

The _Database-Feature VST displays the list of features that are active and/or enabled within the database.

Field Name	Data Type	Description
_DBFeature-ID	INT64	Feature number
_DBFeature_Name	CHARACTER	Name of the database feature
_DBFeature_Enabled	CHARACTER	"1" indicates the feature is enabled; "0" indicates the feature is not enabled
_DBFeature_Active	CHARACTER	"1" indicates this feature is active within the database; "0" indicates this feature is not active

Field Name	Data Type	Description
DBFeature-Res01	CHARACTER	Reserved for future use
DBFeature-Res02	INTEGER	Reserved for future use

Database startup parameters file status (_DbParams)

The _DbParams VST displays information about every database startup parameter.

Field name	Data type	Description
_DbParams-Name	CHARACTER	Parameter name as specified on command line or in a parameter file
_DbParams-Value	CHARACTER	The current value specified for the parameter
_DbParams-dtype	INTEGER	The _field._dtype value
_DbParams-Desc	CHARACTER	Description of the parameter
_DbParams-Msg-Num	INT64	Message number. Number in () written to the `.lg` file or screen
_DbParams-Is-Default	BOOLEAN	Is the current value the default?
_DbParams-Is-Modifiable	BOOLEAN	Is the value modifiable at run time?

Database status (_DbStatus)

The _DbStatus VST displays a wide variety of status data.[34]

Field name	Data type	Description
_DbStatus-AiBlkSize	INTEGER	Number of bytes in an after-image block.
_DbStatus-BiBlkSize	INTEGER	Number of bytes in a before-image block.
_DbStatus-BiClSize	INTEGER	Number of kilobytes in the before-image cluster.
_DbStatus-BiOpen	CHARACTER	Most recent BI file open.
_DbStatus-BiSize	INT64	Logical size of the BI file. You can use this in conjunction with the _DbStatus-BiClSize value to determine the number of clusters in the BI file.
_DbStatus-BiTrunc	CHARACTER	Time since the last truncate.
_DbStatus-CacheStamp	CHARACTER	Schema cache file time stamp.

Field name	Data type	Description
_DbStatus-Changed	INTEGER	Flag indicating if database has changed since last backup.
_DbStatus-ClVersMinor	INTEGER	Client minor version number.
_DbStatus-Codepage	CHARACTER	The value in this field is not maintained. This field will be removed in a future release.
_DbStatus-Collation	CHARACTER	The value in this field is not maintained. This field will be removed in a future release.
_DbStatus-CreateDate	CHARACTER	Date and time of database creation.
_DbStatus-DbBlkSize	INTEGER	Database block size.
_DbStatus-DbVers	INTEGER	Database version number.
_DbStatus-DbVersMinor	INTEGER	Database minor version number.
_DbStatus-EmptyBlks	INT64	Number of empty blocks in the database.
_DbStatus-FbDate	CHARACTER	Most recent full backup.
_DbStatus-FreeBlks	INT64	Number of free blocks in the database.
_DbStatus-HiWater	INTEGER	Database blocks high-water mark.
_DbStatus-IbDate	CHARACTER	Most recent incremental backup.
_DbStatus-IbSeq	INTEGER	Sequence of last incremental backup.
_DbStatus-InitiateDiag	LOGICAL	Set to begin diagnostic data collection immediately
_DbStatus-Integrity	CHARACTER	DbStatus-Integrity enabled flag (`-i`).
_DbStatus-IntFlags	INTEGER	Integrity flags.
_DbStatus-LastLgTruncateTime	CHARACTER	Date-time of the most recent truncation of the database log file
_DbStatus-LastOpen	CHARACTER	Most recent database open.
_DbStatus-LastTable	INTEGER	Highest table number defined.
_DbStatus-LastTran	INTEGER	Last transaction ID.
_DbStatus-MostLocks	INTEGER	Lock table high-water mark.
_DbStatus-NumAreas	INTEGER	Number of areas.

Field name	Data type	Description
_DbStatus-NumLocks	INTEGER	Lock table entries in use.
_DbStatus-NumSems	INTEGER	Number of semaphores.
_DbStatus-PrevOpen	CHARACTER	Previous database open.
_DbStatus-RestrictMode	CHARACTER	The state of any access restrictions set by PROUTIL DBRESTRICT.
_DbStatus-RMFreeBlks	INT64	Number of rm blocks with free space.
_DbStatus-SharedMemVer	INTEGER	Shared memory version number.
_DbStatus-ShmVers	INTEGER	Shared memory version number.
_DbStatus-Starttime	CHARACTER	Time when the database was started.
_DbStatus-State	INTEGER	State of the database.
_DbStatus-Tainted	INTEGER	Damaged database flags.
_DbStatus-TotalBlks	INT64	Number of blocks allocated to the database.
_DbStatus-ZeroStatsDate	CHARACTER	Date/time of the last PROUTIL ZEROSTATS operation on the database.

Database file status (_Filelist)

The _Filelist VST displays the filename, file size, and the size of every database file and extent.[34]

Field name	Data type	Description
_FileList-BlkSize	INTEGER	Block size of the file
_FileList-Extend	INTEGER	Amount of most recent extend in blocks
_FileList-LogicalSz	INTEGER	Logical file size, in blocks
_FileList-Name	CHARACTER	Name of the file
_FileList-Openmode	CHARACTER	Displays the mode in which the file is opened
_FileList-Size	INTEGER	Size of the file

Index statistics (_IndexStat)

The _IndexStat VST displays statistics on the number of accesses to a specific range of indexes.[34]

Field name	Data type	Description
_IndexStat-Blockdelete	INT64	Number of block deletes that have occurred to the index
_IndexStat-Delete	INT64	Number of times delete access has occurred to the index
_IndexStat-Create	INT64	Number of times create access has occurred to the index
_IndexStat-id	INT64	Represents the index number. The `_IndexStat` table is identified by the index number. The `_IndexStat-id` field of the first row of the _IndexStat table is equal to or greater than the value specified by the `-baseindex` startup parameter.
_IndexStat-Read	INT64	Number of times read access has occurred to the index
_IndexStat-OsRead	INT64	Number of index reads from disk since the database was started.
_IndexStat-PartitionId	INTEGER	Partition Id of the index
_IndexStat-Split	INT64	Number of split operations that have occurred to the index

Latch statistics (_Latch)

The _Latch VST displays statistics on latch and latch queue activity.

Field name	Data type	Description
_Latch-Busy	INT64	Latch busy count
_Latch-Hold	INTEGER	User number of the last latch holder
_Latch-HoldTenantId	INTEGER	Tenant ID for the user holding the latch
_Latch-Lock	INT64	Latch lock count
_Latch-LockedT	INTEGER	Used for calculation of duration of lock
_Latch-LockT	INTEGER	Number, in microseconds, the latch was locked
_Latch-Name	CHARACTER	Latch name
_Latch-Owner	INTEGER	User number of the latch owner.
_Latch-Qhold	INTEGER	User number of the last queue holder

Field name	Data type	Description
_LatchQHoldTenantId	INTEGER	Tenant ID for the last queue holder
_Latch-Spin	INTEGER	Latch spin count
_Latch-Type	CHARACTER	Latch type
_Latch-Wait	INT64	Latch wait (nap) count
_Latch-WaitT	INTEGER	Number, in microseconds, of wait or spin time

License management (_License)

The _License VST provides information about the number of users and connections.

Field name	Data type	Description
_Lic-ActiveConns	INTEGER	Number of active connections
_Lic-BatchConns	INTEGER	Number of current batch connections
_Lic-CurrConns	INTEGER	Number of current connections
_Lic-MaxActive	INTEGER	Maximum number of active connections
_Lic-MaxBatch	INTEGER	Maximum number of batch connections
_Lic-MaxCurrent	INTEGER	Maximum number of current connections
_Lic-MinActive	INTEGER	Minimum number of active users
_Lic-MinBatch	INTEGER	Minimum number of batch connections
_Lic-MinCurrent	INTEGER	Minimum number of current connections
_Lic-ValidUsers	INTEGER	Number of valid license users

Lock table status (_Lock)

The _Lock VST Displays the status of the lock table, including the user number, the user name, lock type, RECID number, flags, and chain.

Note: Use caution querying the _Lock table; frequent access can consume large quantities of system resources and negatively impact performance.

Field name	Data type	Description
_Lock-Chain	INTEGER	Chain number

Field name	Data type	Description
_Lock-DomainId	INT64	Domain Id for the user that owns the lock
_Lock-Flags	CHARACTER	Flags for the lock—the flags specify a share lock (S), exclusive lock (X), a lock upgraded from share to exclusive(U), a lock in limbo (L), a queued lock(Q), a lock kept across transaction end boundary (K), a lock is part of a JTAtransaction (J), a lock is in create mode for JTA (C), or a lockwait timeout has expired on this queued lock (E)
_Lock-Name	CHARACTER	User name
_Lock-PartitionId	INTEGER	Partition ID of the locked entry
_Lock-RecId	INT64	RECID of the record and lock table entry
_Lock-Table	INTEGER	Table name
_Lock-TenantId	INTEGER	Tenant ID for the user that owns the lock
_Lock-Trans-Flags	CHARACTER	Transaction flags; see Lock Table on page 575
_Lock-Trans-State	CHARACTER	Transaction state; see PROMON Record Locking Table option on page 554
_Lock-TransId	INTEGER	Transaction id of the lock
_Lock-Tty	CHARACTER	Device from which the user has connected to the database; see also _Connect._Connect-Device
_Lock-Type	CHARACTER	Type of lock (for example, REC, RGET)
_Lock-Usr	INTEGER	User number of the user who owns the lock entry

Lock request (_LockReq)

The _LockReq VST displays information about lock requests, including user name and number, record locks and waits, schema locks and waits, and transaction locks and waits.

Field name	Data type	Description
_LockReq-ExclFind	INT64	Number of exclusive lock finds acquired by the user
_LockReq-Name	CHARACTER	User name
_LockReq-Num	INTEGER	User number
_LockReq-RecLock	INT64	Number of record locks acquired by the user

Field name	Data type	Description
_LockReq-RecWait	INT64	Number of times the user had to wait for a record lock
_LockReq-SchLock	INT64	Number of schema locks acquired by the user
_LockReq-SchWait	INT64	Number of times the user had to wait for a schema lock
_LockReq-ShrFind	INT64	Number of share locks released due to not satisfying a query
_LockReq-TenantId	INTEGER	Tenant ID of the user
_LockReq-TrnLock	INT64	Number of times a deleted record is protected until transaction end
_LockReq-TrnWait	INT64	Number of times a user waited for a record marked for deletion by an active transaction

Logging (_Logging)

The _Logging VST includes before-image and after-image logging information and two-phase commit status.

Field name	Data type	Description
_Logging-2PC	CHARACTER	"Yes" indicates that two-phase commit is enabled. "No" indicates that two-phase commit is disabled.
_Logging-2PCNickName	CHARACTER	Nickname used to identify the coordinator database. You specify a nickname when you enable two-phase commit.
_Logging-2PCPriority	INTEGER	Priority for the coordinator database. You specify the priority when you enable two-phase commit.
_Logging-AiBegin	CHARACTER	Date of the last AIMAGE BEGIN command.
_Logging-AiBlkSize	INTEGER	Size of the after-image block. You can change the AI block size. This reduces I/O rates on disks that have AI files.
_Logging-AiBuffs	INTEGER	Number of buffers in the AI buffer pool. You can change the number of AI buffers with the After-image Buffers (`-aibufs`) startup parameter. The default value is 1.
_Logging-AiCurrExt	INTEGER	Current after-image extent.
_Logging-AiExtents	INTEGER	Number of AI files or extents.

Field name	Data type	Description
_Logging-AiGenNum	INTEGER	Generation number of the current AI extent.
_Logging-AiIO	CHARACTER	After-image I/O. "Reliable" indicates that synchronous or direct I/O is being used for AI writes. "Buffered" indicates that buffered I/O is being used for AI writes.
_Logging-AiJournal	CHARACTER	"YES" indicates that after-imaging is enabled; "NO" indicates that after-imaging is disabled.
_Logging-AiLogSize	INTEGER	Number of kilobytes in the AI log file, if the AI extent in use is variable length. If the AI extent in use is fixed length, the value is 0.
_Logging-AiNew	CHARACTER	Date of the last AIMAGE NEW command.
_Logging-AiOpen	CHARACTER	Date when the AI log was last opened.
_Logging-AISequence	INT64	The database AI sequence number (of the current AI file)
_Logging-BiBlkSize	INTEGER	BI block size.
_Logging-BiBuffs	INTEGER	Number of BI buffers.
_Logging-BiBytesFree	INTEGER	Number of free bytes remaining in the current BI cluster.
_Logging-BiClAge	INTEGER	Period of time that must pass before a BI cluster is reused. This period ensures that database blocks flushed at checkpoint are moved from the UNIX buffers on disk. When this occurs, the transaction is durably recorded on disk.
_Logging-BiClSize	INTEGER	BI cluster size.
_Logging-BiExtents	INTEGER	Number of BI extents.
_Logging-BiFullBuffs	INTEGER	Number of full BI buffers.
_Logging-BiIO	CHARACTER	One of the following: "Reliable" indicates that synchronous or direct I/O is being used for BI writes; "BUFFERED" indicates that buffered I/O is being used for BI writes.
_Logging-BiLogSize	INTEGER	Number of kilobytes in the BI file.
_Logging-BIClusterHWM	INT64	The highest cluster ever used
_Logging-CommitDelay	INTEGER	Current value of the Delayed BI File Write (`-Mf`) parameter.

Field name	Data type	Description
_Logging-CrashProt	INTEGER	"Yes" indicates that crash protection is enabled; "No" indicates that crash protection is disabled.
_Logging-CurrBICluster	INT64	The current BI Cluster
_Logging-LastCkp	CHARACTER	Time of the last checkpoint.

Master block (_MstrBlk)

The _MstrBlk VST displays before-image and after-image information about the master block and other master block status.[34]

Field name	Data type	Description
_MstrBlk-AiBlksize	INTEGER	Number of bytes in an AI block
_MstrBlk-BiBlksize	INTEGER	Number of bytes in a BI block
_MstrBlk-BiOpen	CHARACTER	Date and time the BI file was last opened
_MstrBlk-BiPrev	CHARACTER	Previous value of _MstrBlk-biopen
_MstrBlk-BiState	INTEGER	Current state of BI file
_MstrBlk-Cfilnum	INTEGER	Highest file number currently defined in the database
_MstrBlk-Crdate	CHARACTER	Date and time of database creation
_MstrBlk-Dbstate	INTEGER	Current state of the database
_MstrBlk-Dbvers	INTEGER	Database version number
_MstrBlk-Fbdate	CHARACTER	Date and time of the last full backup
_MstrBlk-Hiwater	INTEGER	Database blocks high-water mark
_MstrBlk-Ibdate	CHARACTER	Date and time of most recent incremental backup
_MstrBlk-Ibseq	INTEGER	Sequence of last incremental backup
_MstrBlk-Integrity	INTEGER	Database integrity enabled flag
_MstrBlk-Lasttask	INTEGER	Last transaction ID
_MstrBlk-Oppdate	CHARACTER	Date and time of previous database open
_MstrBlk-Oprdate	CHARACTER	Date and time of most recent database open
_MstrBlk-Rlclsize	INTEGER	Current BI cluster size

Field name	Data type	Description
_MstrBlk-Rltime	CHARACTER	Time since last before-image file truncate
_MstrBlk-Tainted	INTEGER	Database damaged flags
_MstrBlk-Timestamp	CHARACTER	Schema cache file time stamp
_MstrBlk-Totblks	INT64	Number of blocks allocated to the database

User connection (_MyConnection)

The _MyConnection VST provides information about a user and the number of private read-only buffers allowed and in use.

Table 116:

Field name	Data type	Description
_MyConn-NumSeqBuffers	INTEGER	Number of private read-only buffers currently allowed
_MyConn-Pid	INTEGER	The user's process ID
_MyConn-ServerPid	INTEGER	The process ID of the server the user is connected to
_MyConn-ServerTid	INTEGER	The thread ID of the server thread the user is connected to
_MyConn-TenantId	INTEGER	The user's tenant ID
_MyConn-UsedSeqBuffers	INTEGER	Number of private read-only buffers currently in use
_MyConn-UserId	INTEGER	The user's user number
_MyConn-UserMisc	CHARACTER	User updatable field.

Resource queue statistics (_Resrc)

The _Resrc VST displays statistics on resource queue utilization.

Field name	Data type	Description
_Resrc-Name	CHARACTER	Resource queue name
_Resrc-Lock	INT64	Resource queue lock count

Field name	Data type	Description
_Resrc-Wait	INT64	Resource queue wait count
_Resrc-Time	INT64	Resource queue wait time

Segments (_Segments)

The _Segments VST reports the segment number, segment size, and the number of free and used segments.

Field name	Data type	Description
_Segment-ByteFree	INT64	Number of free bytes in this shared memory segment
_Segment-BytesUsed	INT64	Number of bytes used in this shared memory segment
_Segments-SegId	INTEGER	Segment ID
_Segments-SegSize	INT64	Segment size

Servers (_Servers)

The _Servers VST displays status of each OpenEdge server running on the system, including the server number, process ID, and type; the protocol used; the number of logins and current users; the maximum number of users; and the server's port number.

If the server is a broker (_Server-Pid = _Server-Broker-Pid), then the _SrvParam* fields contain the values of the broker-specific startup parameters.

Field name	Data type	Description
_Server-Broker-Pid	INTEGER	Process ID of the owning broker
_Server-CurrUsers	INTEGER	Number of current users connected to this server
_Server-Logins	INT64	Number of established logins
_Server-MaxUsers	INTEGER	Maximum number of logins available on the server
_Server-Num	INTEGER	Server number
_Server-PendConn	INTEGER	Number of pending connections for this server
_Server-Pid	INTEGER	Process ID of the server
_Server-PortNum	INTEGER	TCP/IP port number of the server
_Server-Protocol	CHARACTER	Protocol supported by the server
_Server-Type	CHARACTER	Server type

Field name	Data type	Description
_SrvParam-Desc	CHARACTER[ ]	Description of the parameter
_SrvParam-dtype	INTEGER[ ]	The _field._dtype value
_SrvParam-Is-Default	LOGICAL[ ]	Is the current value the default?
_SrvParam-Is-Modifiable	LOGICAL[ ]	Is the value modifiable at run time?
_SrvParam-Msg-Num	INT64[ ]	Message number. Number in () written to the `.lg` file or screen
_SrvParam-Name	CHARACTER[ ]	Parameter name as specified on the command line or in a parameter file
_SrvParam-Value	CHARACTER[ ]	The current value specified for the parameter

SQL query plan (_SQL_Qplan)

For SQL only — the _SQL_Qplan VST contains query plans for the last 10 queries that were executed. See *OpenEdge Data Management: SQL Development* for details on the content and use of this VST.

Table statistics (_TableStat)

The _TableStat VST displays statistics on the number of accesses to a specific range of tables.[34]

Field name	Data type	Description
_TableStat-Create	INT64	Number of times create access has occurred to the table
_TableStat-Delete	INT64	Number of times delete access has occurred to the table
_TableStat-Read	INT64	Number of times read access has occurred to the table
_TableStat-id	INT64	Represents the table number. The `_TableStat` table is identified by the table number. The `_TableStat-id` field of the first row of the `_TableStat` table is equal to or greater than the value specified by the `-basetable` startup parameter.
_TableStat-OSRead	INT64	Number of table reads from disk since the database was started; only valid for tables in a Type II area
_TableStat-PartitionID	INTEGER	Partition Id of the table
_TableStat-Update	INT64	Number of times update access has occurred to the table

Transaction (_Trans)

The _Trans VST includes information such as transaction number, state, start time, duration, user number, coordinator name, and transaction.[34]

Field name	Data type	Description
_Trans-BIRecReads	INT64	Number of notes read for the current transaction
_Trans-BIRecWrites	INT64	Number of notes written for the current transaction
_Trans-Coord	CHARACTER	Name of the coordinator database
_Trans-CoordTx	INTEGER	Current coordinator transaction number
_Trans-Counter	INTEGER	Transaction count
_Trans-Duration	INTEGER	Number of seconds the transaction required to complete
_Trans-Flags	CHARACTER	Transaction flags
_Trans-JTA-State	CHARACTER	State of the current JTA transaction. Values include: • JTA-Active • JTA-Idle • JTA-Prepared • JTA-RollbackOnly • JTA-NonExistent
_Trans-Flags	CHARACTER	
_Trans-Misc	INTEGER	Miscellaneous information
_Trans-Num	INTEGER	Transaction number
_Trans-State	CHARACTER	Transaction state
_Trans-TenantID	INTEGER	Tenant ID of the user that is in the transaction
_Trans-Txtime	CHARACTER	Transaction start time
_Trans-Usrnum	INTEGER	User number of the user running the distributed transaction
_Trans-XID	CHARACTER	Global transaction ID for the JTA transaction

Transaction end lock statistics (_TxeLock)

The _TxeLock VST includes statistics about TransactionEnd Locks.

Field name	Data type	Description
_Txe-Locks	INT64	Total number of times that transaction end lock is requested
_Txe-Lockss	INT64	Number of concurrently held locks (only valid for update and commit locks)
_Txe-Time	INTEGER	Not yet implemented. Always "?"
_Txe-Type	CHARACTER	Type of transaction end lock, such as update or commit
_Txe-Waits	INT64	Number of times a transaction end lock request has been queued; how many times you had to wait
_Txe-Waitss	INT64	Number of concurrently queued locks (only valid for update and commit locks)
_Txe-Wait-Time	INTEGER	Not yet implemented. Always "?"

User index activity (_UserIndexStat)

Displays statistics on the number of accesses to a specific range of indexes by user.[34]

Field name	Data type	Description
_UserIndexStat-Conn	INTEGER	User number
_UserIndexStat-Num	INTEGER	Index number
_UserIndexStat-blockdelete	INT64	Number of block deletes that have occurred to the index by the specified user
_UserIndexStat-create	INT64	Number of times create access has occurred to the index by the specified user
_UserIndexStat-delete	INT64	Number of times delete access has occurred to the index by the specified user
_UserIndexStat-PartitionID	INTEGER	Partition Id of the index
_UserIndexStat-read	INT64	Number of times read access has occurred to the index by the specified user
_UserIndexStat-TenantID	INTEGER	Tenant ID of the user
_UserIndexStat-split	INT64	Number of split operations that have occurred to the index by the specified user

Database input/output (_UserIO)

The _UserIO VST displays information about the database input/output operations, including user number and name and the number of accesses, reads, and writes.[34]

Field name	Data type	Description
_UserIO-AiRead	INT64	Number of AI read operations performed by the process
_UserIO-AiWrite	INT64	Number of AI write operations performed by the process
_UserIO-BiRead	INT64	Number of BI read operations performed by the process
_UserIO-BiRecRead	INT64	Number of BI notes read
_UserIO-BiRecWrite	INT64	Number of BI notes written
_UserIO-BiWrite	INT64	Number of BI write operations performed by the process
_UserIO-DbAccess	INT64	Number of database access operations performed by the process
_UserIO-DbRead	INT64	Number of database read operations performed by the process
_UserIO-DbWrite	INT64	Number of database write operations performed by the process
_UserIO-Name	CHARACTER	User name of the process
_UserIO-TenantId	INTEGER	Tenant ID of the user
_UserIO-Usr	INTEGER	User number of the process

Record locking table (_UserLock)

The _UserLock VST displays the contents of the record locking table, such as user name, chain, number, record ID, lock type, and flags. _UserLock reports the first 512 entries for a particular user for each request. If a single user has more than 512 locks, the locks will not be displayed in the _UserLock array.

Note: Use caution when querying _UserLock; doing so can consume large quantities of system resources. Access _UserLock through database monitoring tools written in ABL and limit querying execution paths that are frequently accessed.

Field name	Data type	Description
_UserLock-Chain	INTEGER	Type of chain should always be REC, the record lock chain
_UserLock-Flags	CHARACTER	Any of four possible types of flags: L (LIMBO lock), P (PURGED lock entry), Q (QUEUED lock request), and U (UPGRADE request)Flags for the lock—the flags specify a share lock (S), exclusive lock (X), a lock upgraded from share to exclusive (U), a lock in limbo (L), a queued lock (Q), a lock kept across transaction end boundary (K), a lock is part of a JTA transaction (J), a lock is in create mode for JTA (C), or a lock wait timeout has expired on this queued lock (E)
_UserLock-HWM	INTEGER	Maximum number of locks ever held by this user
_UserLock-Misc	INTEGER	Miscellaneous information
_UserLock-Name	CHARACTER	User name of the process owning the lock
_UserLock-Recid	INT64	Record ID for the record being locked
_UserLock-Table	INTEGER	Table number
_UserLock-TenantId	INTEGER	Tenant ID of the user
_UserLock-Type	CHARACTER	One of the three lock types: EXCL (EXCLUSIVE-LOCK), SHR (SHARE-LOCK), or NOLK (NO-LOCK)
_UserLock-Usr	INTEGER	User number of the process owning the lock

User status (_UserStatus)

The _UserStatus VST displays the progress of PROUTIL and PROBKUP utilities.

Field name	Data type	Description
_UserStatus-Counter	INTEGER	Count blocks accessed, records moved, tables moved, or indexes compacted, etc
_UserStatus-ObjectId	INTEGER	The table or index worked on
_UserStatus-ObjectType	INTEGER	The type of object worked on
_UserStatus-Operation	CHARACTER	Name of the online utility for which status is being monitored
_UserStatus-State	INTEGER	The state the utility is in. See **_UserStatus-State descriptions** for descriptions of utility states)

Field name	Data type	Description
_UserStatus-Target	INTEGER	The end value of the counter (if known)
_UserStatus-TenantId	INTEGER	Tenant ID of the user
_UserStatus-UserId	INT64	User number

Table 117: _UserStatus-State descriptions

_UserStatus-State value	Utility	Description
11	PROUTIL	Attempting to get a table lock
12	PROUTIL	Attempting to get an administration lock
13	PROUTIL	Attempting to lock a storage object
24	PROUTIL TABLEMOVE	Copying records from one area to another
25	PROUTIL TABLEMOVE	Creating a new, alternate index
26	PROUTIL TABLEMOVE	Removing records from the source area after copying from source to target
27	PROUTIL TABLEMOVE	Removing old indexes
31	PROUTIL IDXMOVE	Creating a new index
32	PROUTIL IDXMOVE	Removing old index
41	PROUTIL IDXCOMPACT	Scanning the index delete chain
42	PROUTIL IDXCOMPACT	Compacting a non-leaf index block
43	PROUTIL IDXCOMPACT	Compacting a leaf index block
51	PROUTIL IDXFIX	Scanning the record manager blocks in a database
52	PROUTIL IDXFIX	Scanning the index blocks
53	PROUTIL IDXFIX	Comparing index dbkeys for chosen indexes
54	PROUTIL IDXFIX	Building the target indexes from the source index
60	PROBKUP	Initializing area descriptors
61	PROBKUP	Backing up active before-image clusters
62	PROBKUP	Estimating size of backup files
63	PROBKUP	Flushing the buffer pool and before-image buffers

_UserStatus-State value	Utility	Description
64	PROBKUP	Backing up area object blocks
65	PROBKUP	Backing up the before-image file
66	PROBKUP	Backing up the database blocks per area
70	PROUTIL DBANALYS	Examining the record manager chain
71	PROUTIL DBANALYS	Examining the index delete chain
72	PROUTIL DBANALYS	Examining the free chain
73	PROUTIL DBANALYS	Examining each database block
74	PROUTIL DBANALYS	Gathering fragmentation statistics
80	PROUTIL DUMP	Validating user input
81	PROUTIL DUMP	Validating column index
82	PROUTIL DUMP	Validating column data type
83	PROUTIL DUMP	Dumping records from the specified table using the specified index
84	PROUTIL DUMP	Dumping records for encrypted dump file by dump specified
85	PROUTIL DUMP	Prompt for and validate pass phrase for encrypted dump file
86	PROUTIL DUMP	Preparing information for encrypted dump file header
87	PROUTIL DUMP	Reserved for future use.
88	PROUTIL DUMP	Validating input
89	PROUTIL DUMP	Dumping records by binary dump (non-encrypted dump file)
90	PROUTIL IDXMOVE	Validating user input
91	PROUTIL IDXMOVE	Validating target area
92	PROUTIL IDXMOVE	Attempting to get a table lock
93	PROUTIL IDXMOVE	Moving the index blocks from source to target areas
100	PROUTIL IDXFIX	Creating new indexes for schema index rebuilds

_UserStatus-State value	Utility	Description
101	PROUTIL IDXFIX	Scanning all blocks with index blocks of interest
102	PROUTIL IDXFIX	Scanning all blocks with index or record manager blocks of interest
103	PROUTIL IDXFIX	Scanning all blocks to reconstruct the indexes
104	PROUTIL IDXFIX	Rebuilding indexes from the sort files
105	PROUTIL IDXFIX	Activating every rebuilt index
106	PROUTIL IDXFIX	Deleting old schema indexes
110	PROUTIL TABLEMOVE	Validating user input
111	PROUTIL TABLEMOVE	Validating target area
112	PROUTIL TABLEMOVE	Attempting to get a table lock
113	PROUTIL TABLEMOVE	Looking for inactive indexes
114	PROUTIL TABLEMOVE	Deleting template record from source area
115	PROUTIL TABLEMOVE	Moving table from source to target area
116	PROUTIL TABLEMOVE	Creating template record in target area
117	PROUTIL IDXACTIVATE	Validating command line input
118	PROUTIL IDXACTIVATE	Building index entries phase
119	PROUTIL IDXACTIVATE	Lock the index records of the index(s) being activated
120	PROUTIL IDXACTIVATE	Activating the index in process
121	PROUTIL IDXACTIVATE	Locking the _Index record of the index being activated
122	PROUTIL IDXCHECK	Locking the database schema to support the check process
123	PROUTIL IDXCHECK	Locking the _Index records of indexes being checked from other index utilities
124	PROUTIL IDXCHECK	Locking the table of the index being checked so its data will not change during check.
125	PROUTIL IDXCHECK	Examining the index blocks of an index for and internal validation pass

_UserStatus-State value	Utility	Description
126	PROUTIL IDXCHECK	Examining an index's key entries for internal consistency
127	PROUTIL EPOLICY	Obtaining the encrypted object's internal information for later processing
128	PROUTIL EPOLICY	Scanning the encrypted blocks of an object
129	PROUTIL EPOLICY	Scanning and raising an encrypted object's blocks to a new policy level
130	PROUTIL EPOLICY	Changing an object's encrypted policy's attributes
131	PROUTIL DISABLEENCRYPTION	Attempting to disable AI encryption
132	PROUTIL DISABLEENCRYPTION	Attempting to disable BI encryption
133	PROUTIL DISABLEENCRYPTION	Executing update phase of Disable Encryption
134	PROUTIL DISABLEENCRYPTION	Executing decrypt phase of Disable Encryption
135	PROUTIL DISABLEENCRYPTION	Disabling feature bit of Disable Encryption
136	PROUTIL IDXDEACTIVATE	An attempt to deactivate an index or indexes is in progress
137	PROUTIL IDXDEACTIVATE	Validating the command line input
138	PROUTIL IDXDEACTIVATE	Acquiring the index locks to block other utilities from operating on the index(s)
139	PROUTIL IDXDEACTIVATE	Locking the _Index records of the deactivating index
140	PROUTIL IDXDEACTIVATE	Locking the _StorageObject records of the deactivating index
141	PROUTIL IDXDEACTIVATE	Reserved for future use.
142	PROUTIL DEALLOCATE	Locking the Schema records to support the deallocation process
143	PROUTIL DEALLOCATE	Locking the _Tenant records to support the deallocation process
144	PROUTIL DEALLOCATE	Locking the table or tables involved in the deallocation process

_UserStatus-State value	Utility	Description
145	PROUTIL DEALLOCATE	Locking the _Index record of the index or indexes involved in the deallocation process
146	PROUTIL DEALLOCATE	Locking the LOB object or objects involved in the deallocation process
147	PROUTIL DEALLOCATE	Changing internal object block values for objects involved in the deallocaton process
148	PROUTIL DEALLOCATE	Wait point encountered because processes are blocking the deallocation process.
149	PROUTIL DEALLOCATE	Releasing all locks acquired by the deallocation process
150	PROUTIL MTIDXBUILD	Obtaining admin lock for MTIDXBUILD
151	PROUTIL MTIDXBUILD	Recreating objects in MTIDXBUILD
152	PROUTIL MTIDXBUILD	Building index tree for MTIDXBUILD
153	PROUTIL DATAMOVE	Validating input for DATAMOVE
154	PROUTIL DATAMOVE	Validating target database for DATAMOVE
155	PROUTIL DATAMOVE	Locking table partition for DATAMOVE
156	PROUTIL DATAMOVE	Checking for inactive indexes in DATAMOVE
157	PROUTIL DATAMOVE	Moving data in DATAMOVE
158	PROUTIL DATAMOVE	Moving a sequence in DATAMOVE
159	PROUTIL DATAMOVE	The table instance is being deallocated
160	PROUTIL DATAMOVE	Comparing schemas in DATAMOVE
161	PROUTIL DATAMOVE	Updating the table instance storage object's object state
162	PROUTIL DATAMOVE	Unlocking the table partition lock
1100	PROSHUT	Waiting for user input

User table activity (_UserTableStat)

The _UserTableStat VST displays statistics on the number of accesses to a specific range of tables by user.[34]

Field name	Data type	Description

_UserTableStat-Conn	INTEGER	User number
_UserTableStat-Num	INTEGER	Table number
_UserTableStat-create	INT64	Number of times create access has occurred to the table by the specified user
_UserTableStat-delete	INT64	Number of times delete access has occurred to the table by the specified user
_UserTableStat-PartitionId	INTEGER	Partition Id of the user
_UserTableStat-read	INT64	Number of times delete access has occurred to the table by the specified user
_UserTableStat-TenantId	INTEGER	Tenant Id of the user
_UserTableStat-update	INT64	Updates the table

Index

2PHASE BEGIN qualifier
  PROUTIL utility *259*, *638*
2PHASE COMMIT qualifier
  PROUTIL utility *261*, *625*, *638*
2PHASE END qualifier
  PROUTIL utility *260*, *639*
2PHASE MODIFY qualifier
  PROUTIL utility *639*
2PHASE RECOVER qualifier
  PROUTIL utility *261*, *625*, *640*
4GL tools *412*

A

Activity option
  PROMON utility *556*
ADD qualifier
  PROSTRCT utility *329–331*, *787*
ADDONLINE qualifier
  PROSTRCT utility *788*
-adminport startup parameter *480*, *482*
AdminServer
  hosts *822*
  ports *75*, *482*, *822*, *830*
AdminServer Port (-adminport) startup parameter *480*, *482*
After-image (AI)
  areas *141*, *842*
  files
    archiving *147*
    managing *144*
    marking as empty *149*
    monitoring *144*
    switching *146*
  log *232*
After-image Buffers (-aibufs) startup parameter *463*, *474*, *484*
After-image extents
  backing up *89*
  backups *93*
  on-line backups *93*
After-image File Management Archive Directory Create (-aiarcdircreate) startup parameter *483*
After-image File Management Archive Directory list (-aiarcdir) startup parameter *483*
After-image File Management Archive Interval (-aiarcinterval) startup parameter *484*
After-image Stall (-aistall) startup parameter *146*, *474*, *485*
After-image writers (AIWs) *463*, *484*, *517*
After-imaging
  disabling *161*, *166*, *654*, *805–806*
  enabling *142*, *232*, *806*, *815*
  roll forward recovery *123*
-aiarcdir startup parameter *483*
-aiarcdircreate startup parameter *483*
AIARCHIVE EXTENT qualifier
  RFUTIL utility *803*
AIARCHIVE NEXTEXTENT qualifier
  RFUTIL utility *804*
AIARCHIVER DISABLE qualifier
  RFUTIL utility *801*
AIARCHIVER ENABLE qualifier
  RFUTIL utility *801*
AIARCHIVER END qualifier
  RFUTIL utility *802*
AIARCHIVER SETARCDIR qualifier
  RFUTIL utility *802*
AIARCHIVER SETARCINTERVAL qualifier
  RFUTIL utility *803*
-aiarcinterval startup parameter *484*
-aibufs startup parameter *463*, *474*, *484*
AIMAGE AIOFF qualifier
  RFUTIL utility *146*, *166*, *805*
AIMAGE BEGIN qualifier
  RFUTIL utility *142*, *805–806*, *815*
AIMAGE END qualifier
  RFUTIL utility *166*, *806*
AIMAGE EXTENT EMPTY qualifier
  RFUTIL utility *149*, *232*, *807*
AIMAGE EXTENT FULL qualifier
  RFUTIL utility *144*, *149*, *232*, *807–808*
AIMAGE EXTENT LIST qualifier
  RFUTIL utility *144*, *149*, *807–808*
AIMAGE EXTRACT qualifier
  RFUTIL utility *809*
AIMAGE NEW qualifier
  RFUTIL utility *146*, *809*
AIMAGE QUERY qualifier
  RFUTIL utility *810*
AIMAGE SCAN qualifier
  RFUTIL utility *811*
AIMAGE TRUNCATE qualifier
  RFUTIL utility *812*
-aistall startup parameter *146*, *474*, *485*
AIVERIFY qualifier *813–814*
AppServer *830*
Area consistency check (-AreaCheck) startup parameter *485*
-AreaCheck startup parameter *485*
Areas, See Storage areas

Autonomous Schema Update 532
Asynchronous page writers (APWs)
improving performance 499
Asynchronous replication 230
AUDITARCHIVE qualifier
PROUTIL utility 640
AUDITLOAD qualifier
PROUTIL utility 643
AUDITRECONFIG qualifier
PROUTIL utility 644
-SQLAutoStats 533
Authorization keys 673, 759
sqltruncatetoolarge 532
Authorized Data Truncation (-SQLTruncateTooLarge) startup parameter 532
Auto Server (-m1) startup parameter 478, 509
Autonomous Update Statistics 533

B

-B startup parameter 474, 486, 499, 647
-B2 startup parameter 486
Backup Counter 99
Backups
after-imaging 95
after-imaging (AI) 148
archiving 100
before-image files 89
developing strategy 122
full 98–99, 107, 794, 832, 844
important files 89
incremental 99, 105, 832, 844
media 95
offline 101, 105
online 98, 102, 107, 808
operating system 104, 815
restoring 113
structure files 89
testing 100
types
off-line 93
on-line 93
unscheduled 95
verifying 104, 110, 844
Base Index (-baseindex) startup parameter 479, 487
Base Table (-basetable) startup parameter 479, 487
Base user index (-baseuserindex) startup parameter 488
Base user table (-baseusertable) startup parameter 488
-baseindex startup parameter 479, 487
-basetable startup parameter 479, 487
-baseuserindex startup parameter 488
-baseusertable startup parameter 488
Before-image (BI)
areas 842
blocks 776
clusters 776
Before-image (BI) (continued)
files
truncating 135, 646, 745, 775
Before-image Block Size (-biblocksize) startup parameter 776
Before-image Buffers (-bibufs) startup parameter 464, 474, 484, 489
Before-image Cluster Age (-G) startup parameter 474, 501
Before-image Cluster Size (-bi) startup parameter 776
Before-image files
backing up 89
Before-image Threshold (-bithold) startup parameter 465
Before-image writers (BIWs) 464, 489, 517, 868
-bi startup parameter 776
-biblocksize startup parameter 776
-bibufs startup parameter 464, 474, 484, 489
BIGROW qualifier
PROUTIL utility 646
Binary large objects (BLOBs) 420, 689, 739
-bistall startup parameter 474, 489–490
-bithold startup parameter 465, 474, 489–490
BLOBS, See Binary large objects (BLOBs)
Block Access option
PROMON utility 552
Blocking factor 99, 832
Blocks
before-image (BI) 512
free chain 661
index 661, 712, 715
master 795
record 661
size 794
Blocks in Alternate Buffer Pool (-B2) startup parameter 486
Blocks in Database Buffers (-B) startup parameter 474, 486, 499, 647
Brokers
reservations 520
starting 76
stopping 82
Buffer pool 502
Buffered I/O (-r) startup parameter 524
Buffers
before-image (BI) 501
message 500, 514
BUILD INDEXES qualifier
PROUTIL utility 427
BUILDDB qualifier
PROSTRCT utility 789
Bulk Loader description files
creating 430
modifying 431
sample 430
Bulk loading 430, 502, 647

BULKLOAD qualifier
PROUTIL utility *432*, *647*
BUSY qualifier
PROUTIL utility *104*, *648*

C

-cache startup parameter *490*
Case Table (-cpcase) startup parameter *478*, *493*
CDC *396*
CDC cache size (-cdcsize) startup parameter *491*
cdcsize *396*
-cdcsize startup parameter *491*
Century Year Offset (-yy) startup parameter *480*, *540*, *647*
Chains
free *649*
RM *649*
CHANALYS qualifier
PROUTIL utility *649*
Character large objects (CLOBs) *420*, *656*
Character sets *76*, *656*, *658*
Characters
allowed for database names *68*
-classpath startup parameter *480*, *492*
Clients *520*, *525*
CLOBs, See Character large objects (CLOBs)
Cluster Mode (-cluster) startup parameter *492*
-cluster startup parameter *492*
Clusters
before-image (BI) *501*, *646*
CODEPAGE-COMPILER qualifier
PROUTIL utility *493*, *654*
Collation Table (-cpcoll) startup parameter *478*, *493*
Commands
PROAIW *463*
PROAPW *463*
PROBIW *464*
PROQUIET *465*
PROSERVE *76*, *465*, *467*
PROSHUT *82–83*, *463–464*, *468*
PROWDOG *77*, *471*
Configuration Properties File (-properties) startup parameter *480*, *524*
Connecting
over a wide area network *490*
Connection security *173*
CONV1112 qualifier
PROUTIL utility *53*, *654*
CONVCHAR qualifier
PROUTIL utility *656*
Conversion Map (-convmap) startup parameter *478*, *493*
CONVFILE qualifier
PROUTIL utility *658*
-convmap startup parameter *478*, *493*
Coordinator Information option
PROMON utility *625*
-cpcase startup parameter *478*, *493*
-cpcoll startup parameter *478*, *493*
-cpinternal startup parameter *76*, *478*, *493–494*, *632*, *786*, *798*
-cplog startup parameter *478*, *494*, *496*
-cpprint startup parameter *478*, *495–496*
-cprcodein startup parameter *478*, *495*
-cpstream startup parameter *478*, *496*, *632*, *786*, *798*
-cpterm startup parameter *478*, *496–497*
Crash Recovery Status (-crStatus) startup parameter *497*
Crash Recovery Transaction Display (-crTXDisplay) startup parameter *498*
Crashes
system *130*
system., See System crashes
CRC, See Cyclic redundancy check (CRC)
CRC codes *111*
CREATE qualifier
PROSTRCT utility *41–43*, *45–46*, *49*, *232*, *786*, *790*
Creating
incremental data definitions files *415*
structure description (.st) files *43*
-crStatus startup parameter *497*
-crTXDisplay startup parameter *498*
Cyclic redundancy check (CRC) *689*, *739*, *759*

D

Data ownership models *230*
See also Data consolidation model
Data Administration *41–42*, *52*
Data Administration tool *412*, *414*, *422*, *647*, *745*, *825*
Data collision *230–231*
Data compression *99*, *832*
Data consolidation model *230–231*
Data Dictionary *41–42*, *52*
Data Dictionary tool *50*, *412*, *414*, *422*, *647*, *745*, *825*, *841*
Data distribution model *230*
Data extents
backing up *89*
Data Types
limits *70*
Data Values
limits *70*
Database
connecting to
over a wide area network *490*
Database Client Notification *321–322*
Database Client Notification (-usernotifytime) startup parameter *540*
Database consistency check (-DbCheck) startup parameter *498*

Database files
backing up 89
Database limits
block sizes 60
index limits 65
number of sequences 66
number of simultaneous transactions 67
number of users 67
records per block 60
size 67
Database names
characters not used 68
Database Service Communication Area Size (-pica) startup parameter 521
Database Status option
PROMON utility 562
Databases
backing up, See Backups
backups
types., See Backups
backwards compatibility 689, 739
converting 41–42, 53, 654
coordinator 256, 258, 261, 267, 623, 638–639
copying 56
crash recovery 118
creating
Data Administration 52
Data Dictionary 52
Data Dictionary tool 841
PRODB 51
PROSTRCT CREATE 43
damaged 136–138, 795
deleting 58
dumping contents 137
loading 647
maintaining
indexes 343
tables 343
modes
buffered 738
multi-user 648, 709, 805
single-user 648, 709, 798
unbuffered 738
naming conventions for different operating systems 68
nicknames 259
online 792
quiet points, See Quiet points
reconstructing 815
recovering 118, 122, 130–131
recovery plans 123
restoring
full 114
incremental 115
roll-forward recovery 120, 129, 161
schema, See Schema
Databases *(continued)*
source 839
starting 75
stopping 82, 85
table limits 65
target 412, 434, 839, 844
unlocking 137
DATAMOVE qualifier
PROUTIL utility 660
DataServers
ODBC 830
ORACLE 830
DBANALYS qualifier
PROUTIL utility 661, 767
DBAUTHKEY qualifier
PROUTIL utility 673, 759
-DbCheck startup parameter 498
DBIPCS qualifier
PROUTIL utility 674
DBMAN utility 73–74, 528, 822, 825, 830
DBRESTRICT qualifier
PROUTIL utility 675
DBTOOL utility 825
DEALLOCATE qualifier
PROUTIL utility 677
Delayed BI File Write (-Mf) startup parameter 474, 501, 512
Dependency 242
DESCRIBE qualifier
PROUTIL utility 678
Direct I/O (-directio) startup parameter 474, 499
-directio startup parameter 474, 499
Directories
for temporary files 535
DISABLEAUDITING qualifier
PROUTIL utility 681
DISABLEAUTHGATEWAY qualifier
PROUTIL utility 682
DISABLEB2 qualifier
PROUTIL utility 683
DISABLECDC qualifier
PROUTIL utility 683
DISABLEENCRYPTION qualifier
PROUTIL utility 684
DISABLEJTA qualifier
PROUTIL utility 685
DISABLEKEYEVENTS qualifier
PROUTIL Utility 686
DISABLEMULTITENANCY qualifier
PROUTIL Utility 686
DISABLEREADONLYPARTITIONS qualifier
PROUTIL utility 687
DISABLETABLEPARTITIONING qualifier
PROUTIL utility 687
Disk mirroring 102
Disk space 141

Disks
full 133
DISPTOSSCREATELIMITS qualifier
PROUTIL utility 688
DSRUTIL utility 830
DUMP qualifier
PROUTIL utility 419–420, 689
Dumping
auto-connect records 416
binary 418–419, 689, 692
contents 418
definitions 412
field contents 420
sequence definitions 416
sequence values 424
SQL tables 849
SQL view contents 425
table contents 412, 422
user table content 424
Dumping and loading
binary
aborted 428
economizing disk space 434
limitations 412
manually 412
optimizing data access 435–436
overview 412
special purposes 433
DUMPSPECIFIED qualifier
PROUTIL utility 420, 692

E

-enforceMm startup parameter 500
-ecsize startup parameter 499
ENABLEAUDITING qualifier
PROUTIL utility 695
ENABLEAUTHGATEWAY qualifier
PROUTIL utility 696
ENABLEB2 qualifier
PROUTIL utility 697
ENABLECDC qualifier
PROUTIL utility 698
ENABLEENCRYPTION qualifier
PROUTIL utility 699
ENABLEJTA qualifier
PROUTIL utility 701
ENABLEKEYEVENTS qualifier
PROUTIL utility 701
ENABLELARGEKEYS qualifier
PROUTIL utility 702
ENABLEMULTITENANCY qualifier
PROUTIL utility 702
ENABLEREADONLYPARTITIONS qualifier
PROUTIL utility 703
ENABLESTOREDPROC qualifier
PROUTIL utility 703
ENABLETABLEPARTITIONING qualifier
PROUTIL utility 704
Enabling multi-tenancy 351
Encryption cache size (-ecsize) startup parameter 499
Encryption cache size (-secsize) startup parameter 526
Enforce Mm (-enforceMm) startup parameter 480, 500
EPOLICY INFO qualifier
PROUTIL utility 704
EPOLICY MANAGE qualifier
PROUTIL utility 705
EPOLICY SCAN qualifier
PROUTIL utility 707
EPOLICY VIEW qualifier
PROUTIL utility 708
Error-correction blocks 111
Event Level (-evtlevel) startup parameter 454, 474, 500
Event Viewer 453
-evtlevel startup parameter 454, 474, 500
Extents
See also After-image extents
after-image (AI)
busy 144
empty 144
full 144
application data (D<italic>n<default para font>) 45, 790
before-image (BI) 45–46, 490, 770, 790
creating 46
data 794
fixed-length 46, 140, 792
name 792
number 792
schema 45–46, 790
size 792
tokens
after-image (ai) 792
before-image (bi) 792
data (d) 792
transaction log (tl) 792
transaction log (TL) 45, 99, 790, 817, 832
type 792
variable-length 46, 140, 792
See also After-image extents

F

Failover clusters
cluster dependency 242
cluster resource 242
nodes 236
policy 242
resource group 242
Failures
client machine 266

Failures *(continued)*
network communication *266*
power *266*
servers *266*
File handles
maximum number *68*
File size limitations *689*
File tokens *141*
Files
after-image (.ai) *230*, *485*, *738*, *776*, *810–811*
application data (.d<italic>n<default para font>) *738*
backing up *89*
before-image (.bi) *99*, *512*, *738*, *745*, *775–776*, *832*
binary dump (.bd) *419–420*
Bulk Loader Description (.fd) *430–432*, *647*
code page (.cp) *656*
conmgr.properties *74–75*, *467*, *524*, *528*, *822*
conversion map *493*, *654*, *656*
data (.d) *423*, *496*
data definition (.df)
format *414*
incremental *415*
database (.db) *50*, *738*, *794*
database log (.lg) *428*
event log (.lg) *439*
log (.lg) *266*, *489*, *494*, *500*, *532–533*, *817*, *843*
SQL dump (.dsql) *849*, *853*
structure description (.st) *46*, *786*, *792*, *794*
text dump (.d) *422*
Firewalls *513*
Fixed-length extents
on-line backups *93*
Full backup *91*

G

-G startup parameter *474*, *501*
Group Delay (-groupdelay) startup parameter *474*, *501*
-groupdelay startup parameter *474*, *501*

H

-H startup parameter *77*, *480*, *502*, *525*
-hash startup parameter *474*, *502*
Hash Table Entries (-hash) startup parameter *474*, *502*
Hex values *656*
HOLDER qualifier
PROUTIL utility *709*
Host Name (-H) startup parameter *77*, *480*, *502*, *525*

I

-i startup parameter *436*, *474*, *502*
IDXACTIVATE qualifier
PROUTIL utility *710*
IDXANALYS qualifier
PROUTIL utility *346*, *419*, *712*, *726*
IDXBUILD *313*
IDXBUILD qualifier
PROUTIL utility *436–437*, *715*
IDXCHECK *313*
IDXCHECK qualifier
PROUTIL utility *722*
IDXCOMPACT qualifier
PROUTIL utility *344*, *346*, *726*
IDXDEACTIVATE qualifier
PROUTIL Utility *728*
IDXFIX *313*
IDXFIX qualifier
PROUTIL utility *346*, *730*
IDXMOVE qualifier
PROUTIL utility *343–344*, *346*, *734*, *770*
increase startup parameters *474*
increase startup parameters online *736*
Incremental backups *91*
Index consistency check (-IndexCheck) startup parameter *503*
Index Range Size (-indexrangesize) startup parameter *479*, *487*, *504*
-IndexCheck startup parameter *503*
Indexes
activating *715*
compacting *344*, *715*, *726*
corrupt *715*, *722*
deactivating *437*
deleting *775*
moving *344*, *734*
repairing *715*
scanning *730*
-indexrangesize startup parameter *479*, *487*, *504*
Internal Code Page (-cpinternal) startup parameter *76*, *478*, *493–494*, *632*, *786*, *798*
Internet Protocol (-ipver) startup parameter *505*
IOSTATS qualifier
PROUTIL utility *738*
-ipver startup parameter *505*

K

Key Alias (-keyalias) startup parameter *480*, *506*
Key Alias Password (-keyaliaspasswd) startup parameter *480*, *506*
Key store
backing up *89*
Key Store Passphrase (-Passphrase) startup parameter *520*
-keyalias startup parameter *480*, *506*
-keyaliaspasswd startup parameter *480*, *506*

L

-L startup parameter *507*
Large Files *42*
Latches *530*
-LGovernor startup parameter *506*
Limits
  database
    number of simultaneous transactions *67*
    number of users *67*
  database names *68*
  for database names *68*
LIST qualifier
  PROSTRCT utility *50*, *52*, *57*, *232*, *328–332*, *342*, *787*, *792*
-lkhash startup parameter *508*
-lkrela startup parameter *474*
LOAD qualifier
  PROUTIL utility *739*
Load records
  bad *433*
  reconstructing *433*
Loading
  binary *739*
  bulk, See Bulk loading types
  contents *426*, *430*
  definitions
    4GL *426*
    table *425*
  no-integrity mode *436*
  sequence values *430*
  SQL tables *853*
  SQL view contents *429*
  table contents
    Data tool *428*
    PROUTIL *427*
    PROUTIL BULKLOAD *432*
  user table contents *429*
Lock Governor (-LGovernor) startup parameter *506*
Lock Table Entries (-L) startup parameter *474*, *507*
Lock Table Hash Size (-lkhash) startup parameter *508*
Lock wait timeout (SQLLockWaitTimeout) startup parameter *531*
Locking and Waiting statistics option
  PROMON utility *551*
Locks
  exclusive *343*, *507*, *554*, *770*
  intent exclusive *554*
  intent share *554*
  share *344*, *507*, *554*, *734*
  shared on table with intent to set exclusive *554*
-SQLLockWaitTimeout startup parameter *531*
Log (.lg) files
  backing up *89*
Log File Code Page (-cplog) startup parameter *478*, *494*, *496*
Login Governor (-nGovernor) startup parameter *518*
LRU alternate buffer pool force skips (-lru2skips) startup parameter *509*
LRU Force skips (-lruskips) startup parameter *508*
-lru2skips startup parameter *509*
-lruskips startup parameter *508*

M

-Mm startup parameter *514*
-m1 startup parameter *478*, *509*
-m3 startup parameter *478*, *510*, *515*
-Ma startup parameter *480*, *510*, *513*
MARK BACKEDUP qualifier
  RFUTIL utility *815*
-maxAreas startup parameter *510*
Maximum area number (-maxAreas) startup parameter *510*
Maximum Clients per Server (-Ma) startup parameter *480*, *510*, *513*
Maximum Dynamic Server (-maxport) startup parameter *480*, *511*, *513*
Maximum JTA Transactions (-maxxids) startup parameter *511*
Maximum Number of Users (-n) startup parameter *510*, *513*
Maximum Servers (-Mn) startup parameter *480*, *510*, *513–515*
Maximum Servers per Broker (-Mpb) startup parameter *480*, *515*
-maxport startup parameter *480*, *511*, *513*
-maxxids startup parameter *511*
-MemCheck startup parameter *512*
Memory consistency check (-MemCheck) startup parameter *512*
Message Buffer Size (-Mm) startup parameter *480*, *514*
Message Wait (-Nmsgwait) startup parameter *519*
-Mf startup parameter *474*, *501*, *512*
-Mi startup parameter *480*, *510*, *513–514*
Minimum Clients per Server (-Mi) startup parameter *480*, *510*, *513–514*
Minimum Dynamic Server (-minport) startup parameter *480*, *511*, *513*
-minport startup parameter *480*, *511*, *513*
-Mn startup parameter *480*, *510*, *513–515*
Modify Defaults option
  PROMON utility *627*
-Mp startup parameter *480*, *515*
-Mpb startup parameter *480*, *515*
MTIDXBUILD qualifier
  PROUTIL utility *743*
Multi-tenancy
  architecture *348*
  enabling multi-tenancy *351*
  multi-tenant aware utilities *368*
  multi-tenant features *350*

Multi-tenancy *(continued)*
multi-tenant groups *349*
overview *347*
simple tenancy *349*
super-tenant *348*
tenant groups *348*
-mux startup parameter *516*
MVSCH qualifier
PROUTIL utility *54*, *745*
-Mxs startup parameter *516*

N

-n startup parameter *463–464*, *474*, *510*, *513*, *517*
-N startup parameter *480*, *517*
NameServer *525*, *830*
Nap Time Steps (-napstep) startup parameter *518*
-napstep startup parameter *518*
Network Type (-N) startup parameter *480*, *517*
Networks
connecting over wide area *490*
-nGovernor startup parameter *518*
-Nmsgwait startup parameter *519*
No Crash Protection (-i) startup parameter *436*, *474*, *502*
No Session Cache (-nosessioncache) startup parameter *480*, *519*
Node *236*
-nosessioncache startup parameter *480*, *519*
Number of checkpoint statistics (-numcheckpointstats) startup parameter *519*
Number of Users (-n) startup parameter *463–464*, *474*, *517*
-numcheckpointstats startup parameter *519*

O

Off-line backups *93*
-omsize startup parameter *520*
On-line backups
after-image extents *93*

P

-P startup parameter *172*
Parameter file *474*
Parameter File (-pf) startup parameter *467*, *474*, *521*, *841*
PARTITIONMANAGE ALTER qualifier
PROUTIL utility *746*
PARTITIONMANAGE COPY qualifier
PROUTIL utility *748*
PARTITIONMANAGE MERGE qualifier
PROUTIL utility *750*
PARTITIONMANAGE RENAME qualifier
PROUTIL utility *751*
PARTITIONMANAGE SPLIT qualifier
PROUTIL utility *752*
PARTITIONMANAGE TRUNCATE qualifier
PROUTIL utility *753*
PARTITIONMANAGE VIEW qualifier
PROUTIL utility *755*
-Passphrase startup parameter *520*
Password (-P) startup parameter *172*
Passwords
changing *176*
Pathnames
absolute *52*, *57*
relative *52*
Peer-to-peer model *230–231*
-PendConnTime startup parameter *480*, *520*
Pending Connection Time (-PendConnTime) startup parameter *480*, *520*
-pf startup parameter *467*, *474*, *521*, *841*
-pica startup parameter *521*
Pin Shared Memory (-pinshm) startup parameter *474*, *522*
-pinshm startup parameter *474*, *522*
Policy *242*
Prefetch delay (-prefetchDelay) startup parameter *522*
Prefetch factor (-prefetchFactor) startup parameter *522*
Prefetch Num Recs (-prefetchNumRecs) startup parameter *523*
-prefetchDelay startup parameter *522*
-prefetchFactor startup parameter *522*
-prefetchNumRecs startup parameter *523*
PrefetchPriority (-prefetchPriority) startup parameter *523*
-prefetchPriority startup parameter *523*
Primary recovery area
See also Before-image (BI)
size *66*
See also Before-image (BI)
Print Code Page (-cpprint) startup parameter *478*, *495–496*
PROADSV utility *73*, *830*
PROAIW command *463*
PROAPW command *463*
PROBIW command *464*
PROBKUP utility *98–99*, *102*, *146*, *815*, *832*
Procedure library (.pl) files
backing up *89*
PROCOPY utility *41–42*, *50*, *56–57*, *141*, *790*, *839*
PRODB utility *41–42*, *51–52*, *56*, *433*, *790*, *841*
PRODEL utility *58*, *434*, *842*
PROLOG utility *441*, *843*
PROMON utility
aborting a limbo transaction *261*
Activity option *556*, *674*
Block Access option *552*
committing a limbo transaction *261*
Coordinator Information option *261*, *625*
Database Status option *562*

PROMON utility *(continued)*
Locking and Waiting statistics option *551*
main menu *261*, *271*, *544*
Modify Defaults option *627*
R&D menu
Activity Displays *594*, *606*
Adjust Monitor Options *622*
Administrative Functions *614*, *616*
Other Displays *530*, *607*, *610*
Status Displays *566*, *583*
Record Locking Table option *554*
Resolve JTA Transactions option *626*
Resolve Limbo Transaction option *625*
resolve limbo transactions *267*
resolving limbo transactions *261*
Shared Resources option *559*
Shut Down a Database option *564*
Shut Down Database option *468*
transaction control *261*
Transactions Control option *623*
User Control option *545*
-properties startup parameter *480*, *524*
PROQUIET command *102*, *104*, *465*
PROREST utility *100*, *112–113*, *141*, *844*
PROSERVE command *76*, *465*, *467*
PROSERVE utility *77*
PROSHUT command
batch shutdown (-b) *468*
batch shutdown with no active users (-bn) *468*
emergency shutdown (-F) *468*
unconditional batch shutdown (-by) *468*
PROSTRCT utility
ADD qualifier *329–331*, *787*
ADDONLINE qualifier *788*
BUILDDB qualifier *789*
CREATE qualifier *41–43*, *45–46*, *49*, *232*, *786*, *790*
LIST qualifier *50*, *52*, *57*, *232*, *328–332*, *342*, *787*, *792*
REMOVE qualifier *332*, *342*, *792*
REORDER AI qualifier *159*, *793*
REPAIR qualifier *102*, *794*
STATISTICS qualifier *326*, *776*, *794*, *812*
UNLOCK qualifier *795*
Protocols
network *515*
PROUTIL utility
2PHASE BEGIN qualifier *259*, *638*
2PHASE COMMIT qualifier *261*, *625*, *638*
2PHASE END qualifier *260*, *639*
2PHASE MODIFY qualifier *639*
2PHASE RECOVER qualifier *261*, *625*, *640*
AUDITARCHIVE qualifier *640*
AUDITLOAD qualifier *643*
AUDITRECONFIG qualifier *644*
BIGROW qualifier *646*
BUILD INDEXES qualifier *427*
PROUTIL utility *(continued)*
BULKLOAD qualifier *432*, *647*
BUSY qualifier *104*, *648*
CHANALYS qualifier *649*
CODEPAGE-COMPILER qualifier *493*, *654*
CONV1112 qualifier *53*, *654*
CONVCHAR qualifier *656*
CONVFILE qualifier *658*
DATAMOVE qualifier *660*
DBANALYS qualifier *661*, *767*
DBAUTHKEY qualifier *673*, *759*
DBIPCS qualifier *674*
DBRESTRICT qualifier *675*
DEALLOCATE qualifier *677*
DESCRIBE qualifier *678*
DISABLEAUDITING qualifier *681*
DISABLEAUTHGATEWAY qualifier *682*
DISABLEB2 qualifier *683*
DISABLECDC qualifier *683*
DISABLEENCRYPTION qualifier *684*
DISABLEJTA qualifier *685*
DISABLEREADONLYPARTITIONS qualifier *687*
DISABLETABLEPARTITIONING qualifier *687*
DISPTOSSCREATELIMITS qualifier *688*
DUMP qualifier *419–420*, *689*
DUMPSPECIFIED qualifier *420*, *692*
ENABLEAUDITING qualifier *695*
ENABLEAUTHGATEWAY qualifier *696*
ENABLEB2 qualifier *697*
ENABLECDC qualifier *698*
ENABLEENCRYPTION qualifier *699*
ENABLEJTA qualifier *701*
ENABLEKEYEVENTS qualifier *701*
ENABLELARGEKEYS qualifier *702*
ENABLEMULTITENANCY qualifier *702*
ENABLEREADONLYPARTITIONS qualifier *703*
ENABLESTOREDPROC qualifier *703*
ENABLETABLEPARTITIONING qualifier *704*
EPOLICY INFO qualifier *704*
EPOLICY MANAGE qualifier *705*
EPOLICY SCAN qualifier *707*
EPOLICY VIEW qualifier *708*
HOLDER qualifier *709*
IDXACTIVATE qualifier *710*
IDXANALYS qualifier *346*, *419*, *712*, *726*
IDXBUILD qualifier *436–437*, *715*
IDXCHECK qualifier *722*
IDXCOMPACT qualifier *344*, *346*, *726*
IDXFIX qualifier *346*, *730*
IDXMOVE qualifier *343–344*, *346*, *734*, *770*
INCREASETO qualifier *736*
IOSTATS qualifier *738*
LOAD qualifier *739*
MTIDXBUILD qualifier *743*
MVSCH qualifier *54*, *745*
PARTITIONMANAGE ALTER qualifier *746*

PROUTIL utility *(continued)*
PARTITIONMANAGE COPY qualifier *748*
PARTITIONMANAGE MERGE qualifier *750*
PARTITIONMANAGE RENAME qualifier *751*
PARTITIONMANAGE SPLIT qualifier *752*
PARTITIONMANAGE TRUNCATE qualifier *753*
PARTITIONMANAGE VIEW qualifier *755*
RCODEKEY qualifier *759*
SETAREACREATELIMIT qualifier *760*
SETAREATOSSLIMIT qualifier *761*
SETLOBCREATELIMITqualifier *762*
SETLOBTOSSLIMIT qualifier *763*
SETTABLECREATELIMIT qualifier *764*
SETTABLETOSSLIMIT qualifier *766*
TABANALYS qualifier *434–435*, *767*
TABLEMOVE qualifier *343*, *346*, *770*
TPIDXBUILD qualifier *773*
TRUNCATE AREA qualifier *342*, *775*
TRUNCATE BI qualifier *342*, *776*
UPDATESCHEMA qualifier *777*
UPDATEVST qualifier *778*, *864*
VIEWB2 qualifier *778*
WBREAK-COMPILER qualifier *781*
WORD-RULES qualifier *781*
ZEROSTATS qualifier *782*
PROUTIL Utility
DISABLEKEYEVENTS qualifier *686*
DISABLEMULTITENANCY qualifier *686*
IDXDEACTIVATE qualifier *728*
PROWDOG command *77*, *471*, *517*

Q

Quiet points *102*

R

-r startup parameter *524*
R-code *415*
R-code in Code Page (-cprcodein) startup parameter *478*, *495*
RCODEKEY qualifier
PROUTIL utility *759*
Record fragmentation *500*, *514*
Record free chain search depth factor (-recspacesearchdepth) startup parameter *524*
Record Locking Table option
PROMON utility *554*
Recovery
roll-forward *161*, *258*
Recovery Log Threshold (-bithold) startup parameter *474*, *489–490*
Recovery plans
developing *122*
-recspacesearchdepth startup parameter *524*
REMOVE qualifier
PROSTRCT utility *332*, *342*, *792*
REORDER AI qualifier
PROSTRCT utility *159*, *793*
REPAIR qualifier
PROSTRCT utility *102*, *794*
Replication
asynchronous., See Asynchronous replication
DSRUTIL utility *830*
log-based
implementing *232*
schemes *229*
Resolve JTA Transactions option
PROMON utility *626*
Resolve Limbo Transactions option
PROMON utility *625*
Resource *242*
Resource group *242*
RFUTIL utility
AIARCHIVE EXTENT qualifier *803*
AIARCHIVE NEXTEXTENT qualifier *804*
AIARCHIVER DISABLE qualifier *801*
AIARCHIVER ENABLE qualifier *801*
AIARCHIVER END qualifier *802*
AIARCHIVER SETARCDIR qualifier *802*
AIARCHIVER SETARCINTERVAL qualifier *803*
AIMAGE AIOFF qualifier *146*, *166*, *805*
AIMAGE BEGIN qualifier *142*, *805–806*, *815*
AIMAGE END qualifier *166*, *806*
AIMAGE EXTENT EMPTY qualifier *149*, *232*, *807*
AIMAGE EXTENT FULL qualifier *144*, *149*, *232*, *807–808*
AIMAGE EXTENT LIST qualifier *144*, *149*, *807–808*
AIMAGE EXTRACT qualifier *809*
AIMAGE NEW qualifier *146*, *809*
AIMAGE QUERY qualifier *810*
AIMAGE SCAN qualifier *811*
AIMAGE TRUNCATE qualifier *812*
MARK BACKEDUP qualifier *815*
roll forward *161*
ROLL FORWARD qualifier *161*, *815*
ROLL FORWARD RETRY qualifier *816–817*
SEQUENCE qualifier *819*
roll forward *161*
ROLL FORWARD qualifier
RFUTIL utility *161*, *815*
ROLL FORWARD RETRY qualifier
RFUTIL utility *816–817*
Roll-forward recovery
do's and don'ts *123*

S

-ssj startup parameter *533*
-S startup parameter *77*, *480*, *525*

Scan (-scan) startup parameter
PROBKUP utility 93
Schema
moving 745
security 177
Schema Cache File (-cache) startup parameter 490
Schema Lock Wait Queue (-schlockwq) startup parameter 526
-schlockwq startup parameter 526
Secondary Login Broker (-m3) startup parameter 478, 510, 515
-secsize startup parameter 526
Security
administrator 175
connection 173
database 177
operating system 177
schema 177
Semaphore Sets (-semsets) startup parameter 474, 527
Semaphores 527, 530
-semsets startup parameter 474, 527
SEQUENCE qualifier
RFUTIL utility 819
Server Group (-servergroup) startup parameter 480, 528
Server groups 524
Server startup parameters
statistic collection specification 479
Server-side JOIN (-ssj) startup parameter 474
Server-Side Join (-ssj) startup parameter 533
-servergroup startup parameter 480, 528
Servers
accessing 513
remote 509–510
starting 76, 513, 515, 524, 776
stopping 82
Servers per Protocol (-Mp) startup parameter 480, 515
Service Name (-S) startup parameter 77, 480, 525
Session Timeout (-sessiontimeout) startup parameter 480, 528
-sessiontimeout startup parameter 480, 528
SETAREACREATELIMIT qualifier
PROUTIL utility 760
SETAREATOSSLIMIT qualifier
PROUTIL utility 761
SETLOBCREATELIMIT qualifier
PROUTIL utility 762
SETLOBTOSSLIMIT qualifier
PROUTIL utility 763
SETTABLECREATELIMIT qualifier
PROUTIL utility 764
SETTABLETOSSLIMIT qualifier
PROUTIL utility 766
Shared device 236
Shared memory segment size (-shmsegsize) startup parameter 529
Shared Resources option
PROMON utility 559
Shared-memory Overflow Size (-Mxs) startup parameter 474, 516
-shmsegsize startup parameter 529
Shut Down Database option
PROMON utility 564
Single-user Progress
on-line backups 93
Spin Lock Retries (-spin) startup parameter 474, 530
-spin startup parameter 530
SQL rowid identifier (-SQLQuotedRowid) startup parameter 531
SQL Server Java Classpath (-classpath) startup parameter 480, 492
SQLDUMP 849
SQLDUMP utility 849
SQLLOAD 853
SQLLOAD utility 853
-SQLQuotedRowid startup parameter 531
SQLSCHEMA utility 857
-ssl startup parameter 480, 534
SSL (-ssl) startup parameter 480, 534
SSL connections 79
.st files, See Structure description (.st) files
Startup parameter
Database Service Communication Area Size (-pica) 521
Lock Table Hash Size (-lkhash) 508
Lock Wait Timeout (-SQLLockWaitTimeout) 531
Nap Time Steps (-napstep) 518
Storage Object Cache Size (-omsize) 520
User MUX Latches (-mux) 516
startup parameters
increase online 474
Startup parameters
AdminServer Port (-adminport) 480, 482
After-image Buffers (-aibufs) 463, 474, 484
After-image File Management Archive Directory Create (-aiarcdircreate) 483
After-image File Management Archive Directory List (-aiarcdir) 483
After-image File Management Archive Interval (-aiarcinterval) 484
After-image Stall (-aistall) 146, 474, 485
Area consistency check (-AreaCheck) 485
Auto Server (-m1) 478, 509
Autonomous Schema Update (-SQLWidthUpdate) 532
Autonomous Update Statistics (-SQLAutoStats) 533
Base Index (-baseindex) 479, 487
Base Table (-basetable) 479, 487
Base user index (-baseuserindex) 488
Base user table (-baseusertable) 488
Before-image Block Size (-biblocksize) 776
Before-image Buffers (-bibufs) 464, 474, 484, 489

Startup parameters *(continued)*
Before-image Cluster Age (-G) *474*, *501*
Before-image cluster size (-bi) *776*
Before-image Threshold (-bithold) *465*
Blocks in Alternate Buffer Pool (-B2) *486*
Blocks in Database Buffers (-B) *474*, *486*, *499*, *647*
Buffered I/O (-r) *524*
Case Table (-cpcase) *478*, *493*
CDC cache size (-cdcsize) *491*
Century Year Offset (-yy) *480*, *540*, *647*
Cluster Mode (-cluster) *492*
Collation Table (-cpcoll) *478*, *493*
Configuration Properties File (-properties) *480*, *524*
Conversion Map (-convmap) *478*, *493*
Crash Recovery Status (-crStatus) *497*
Crash Recovery Transaction Display (-crTXDisplay) *498*
Database Client Notification (-usernotifytime) *540*
Database consistency check (-DbCheck) *498*
Delayed BI File Write (-Mf) *474*, *501*, *512*
Direct I/O (-directio) *474*, *499*
Encryption cache size (-ecsize) *499*
Enforce Mm (-enforceMm) *480*
Event Level (-evtlevel) *454*, *474*, *500*
Group Delay (-groupdelay) *474*, *501*
Hash Table Entries (-hash) *474*, *502*
Host Name (-H) *77*, *480*, *502*, *525*
Index consistency check (-IndexCheck) *503*
Index Range Size (-indexrangesize) *479*, *487*, *504*
Internal Code Page (-cpinternal) *76*, *478*, *493–494*, *632*, *786*, *798*
Key Alias (-keyalias) *480*
Key Alias Password (-keyaliaspasswd) *480*
Key store passphrase (-Passphrase) *520*
Lock Governor (-LGovernor) *506*
Lock Table Entries (-L) *474*, *507*
Log File Code Page (-cplog) *478*, *494*, *496*
Login Governor (-lGovernor) *518*
LRU alternate buffer pool force skips (-lru2skips) *509*
LRU Force skips (-lruskips) *508*
Maximum Clients per Server (-Ma) *480*, *510*, *513*
Maximum Dynamic Server (-maxport) *480*, *511*, *513*
Maximum JTA Transactions (-maxxids) *511*
Maximum Number of Users (-n) *513*
Maximum Servers (-Mn) *480*, *510*, *513–515*
Maximum Servers per Broker (-Mpb) *480*, *515*
Memory consistency check (-MemCheck) *512*
Message Buffer Size (-Mm) *480*
Message Wait (-Nmsgwait) *519*
Minimum Clients per Server (-Mi) *480*, *510*, *513–514*
Minimum Dynamic Server (-minport) *480*, *511*, *513*
Network Type (-N) *480*, *517*
No Crash Protection (-i) *436*, *474*, *502*
No Session Cache (-nosessioncache) *480*
Number of checkpoint statistics (-numcheckpointstats) *519*
Startup parameters *(continued)*
Number of Users (-n) *463–464*, *474*, *510*, *517*
Parameter File (-pf) *467*, *474*, *521*, *841*
Password (-P) *172*
Pending Connection Time (-PendConnTime) *480*, *520*
Pin Shared Memory (-pinshm) *474*, *522*
Prefetch delay (-prefetchDelay) *522*
Prefetch factor(-prefetchFactor) *522*
Prefetch Num Recs (-prefetchNumRecs) *523*
Prefetch priority (-prefetchPriority) *523*
Print Code Page (-cpprint) *478*, *495–496*
R-code in Code Page (-cprcodein) *478*, *495*
Record free chain search depth factor (-recspacesearchdepth) *524*
Recovery Log Threshold (-bithold) *474*, *489–490*
Schema Lock Wait Queue (-schlockwq) *526*
Secondary Login Broker (-m3) *478*, *510*, *515*
Security cache size (-secsize) *526*
Semaphore Sets (-semsets) *474*, *527*
Server Group (-servergroup) *480*, *528*
Server-side JOIN (-ssj) *474*
Servers per Protocol (-Mp) *480*, *515*
Service Name (-S) *77*, *480*, *525*
Session Timeout (-sessiontimeout) *480*
Shared-memory Overflow Size (-Mxs) *474*, *516*
Spin Lock Retries (-spin) *474*, *530*
SQL Server Java Classpath (-classpath) *480*, *492*
SSL (-ssl) *480*
statistic collection specification *479*
Stream Code Page (-cpstream) *478*, *496*, *632*, *786*, *798*
STS logging level (-stsloggінglevel) *534*
Table consistency check (-TableCheck) *536*
Table Range Size (-tablerangesize) *479*, *487*, *537*
Terminal Code Page (-cpterm) *478*, *496–497*
Threaded Server (-threadedServer) *474*
Threaded Server Stack (-threadedServerStack) *474*
Threshold Stall (-bistall) *474*, *489–490*
Transaction End (TXE) Lock Retry Limit (-TXERetryLimit) *538*
Transaction End (TXE) Retry Limit (-TXERetryLimit) *474*
Truncate Too Large (--SQLTruncateTooLarge) *532*
Truncate Too Large (-SQLTruncateTooLarge) *532*
User ID (-U) *172*
User index range (-userindexrangesize) *539*
User table range (-usertablerangesize) *539*
Statistic Collection startup parameters *479*
STATISTICS qualifier
PROSTRCT utility *326*, *776*, *794*, *812*
Storage areas
after-image (AI) *140*, *790*, *806*
application data (.D<italic>n<default para font>) *45*
application data (D<italic>n<default para font>) *775*, *790*

Storage areas *(continued)*
before-image (BI) *465*
creating *46*
database control (DB) *786*, *789–790*
number of *60*
primary recovery *790*
primary recovery area
size *66*
removing *792*
size of *60*
transaction log (TL) *638*, *790*
Storage Object Cache Size (-omsize) startup parameter *520*
Stream Code Page (-cpstream) startup parameter *478*, *496*, *632*, *786*, *798*
Structure description (.st) files
creating *43*
example *47*
Structure files
backing up *89*
STS logging level (-stsloggінglevel) startup parameter *534*
-stsloggінglevel startup parameter *534*
System crashes *130–131*

T

-threadedServer startup parameter *537*
-T startup parameter *535*
-threadedServerStack startup parameter *538*
TABANALYS qualifier
PROUTIL utility *434–435*, *767*
Table consistency check (-TableCheck) startup parameter *536*
Table partitioning
Altering partition state *379*
Database Administration *387*
Merging partitions *377*
Partition-scoped utilities *387*
PROUTIL PARTITIONMANAGE *376*
Read-only partitions *379*
Renaming partitions *378*
Splitting partitions *376*
Truncating partitions *378*
Viewing partitions *379*
Table Range Size (-tablerangesize) startup parameter *479*, *487*, *537*
-TableCheck startup parameter *536*
TABLEMOVE qualifier
PROUTIL utility *343*, *346*, *770*
-tablerangesize startup parameter *479*, *487*, *537*
Tables
ABL *172*
constant *434*
deleting *775*
freezing *177*
Tables *(continued)*
moving *343*
SQL *173*
unfreezing *177*
word-break *781*
TCP, See Before-image (BI)
Temporary Directory (-T) startup parameter *535*
Terminal Code Page (-cpterm) startup parameter *478*, *496–497*
Threaded Server (-threadedServer) startup parameter *474*
Threaded Server Stack (-threadedServerStack) startup parameter *474*, *538*
Threaded Server(-threadedServer) startup parameter *537*
Threshold Stall (-bistall) startup parameter *474*, *489–490*
Tokens
area name *43*
area number *43*
blocks per cluster *43*
extent size *43*
extent type *43*
records per block *43*
storage area type *43*
TPIDXBUILD qualifier
PROUTIL utility *773*
Transaction End (TXE) Lock Retry Limit (-TXERetryLimit) startup parameter *538*
Transaction End (TXE) Retry Limit startup parameter *474*
Transaction log
area *259–260*, *790*
extents *342*, *844*
files *133–134*
Transaction log (.tl) files
backing up *89*
Transactions
committed *638*
distributed *256*, *261*
limbo
aborted *261*, *625*
case study *267*
committed *261*, *267*, *625*
detecting *261*
resolving *261*, *266*
log files *256*
numbers *256*, *261*
Transactions Control option
PROMON utility *623*
Transmission Control Protocol (TCP) *77*, *83*
TRUNCATE AREA qualifier
PROUTIL utility *342*, *775*
TRUNCATE BI qualifier
PROUTIL utility *342*, *776*
truncatetoolarge *532*
-SQLWidthUpdate *532*

Truncate Too Large 532
Two-phase commit
deactivating 260
disabling 639, 654
enabling 259
-TXERetryLimit startup parameter 538

U

-U startup parameter 172
UNIX parameters
SHMALL 674
UNLOCK qualifier
PROSTRCT utility 795
UPDATESCHEMA qualifier
PROUTIL utility 777
UPDATEVST qualifier
PROUTIL utility 778, 864
User Control option
PROMON utility 545
User ID (-U) startup parameter 172
User index range (-userindexrangesize)startup parameter 539
User MUX Latches (-mux) startup parameter 516
User reports 176
User table range (-usertablerangesize)startup parameter 539
-userindexrangesize startup parameter 539
-usernotifytime startup parameter 540
Users
adding 174
deleting 176
IDs 170
local 517
OpenEdge 517
passwords 170–171
remote 510, 513, 515, 517, 525
unauthorized 172
validating 172
_UserStatus virtual system table 344, 346, 712, 722, 726, 734, 770
-usertablerangesize startup parameter 539
Utilities
DBMAN 73–74, 528, 822, 825, 830
DBTOOL 825
DSRUTIL 830
PROADSV 73, 830
PROBKUP 98–99, 102, 146, 815, 832
PROCOPY 41–42, 50, 56–57, 141, 790, 839
PRODB 41–42, 51–52, 56, 433, 790, 841
Utilities *(continued)*
PRODEL 58, 434, 842
PROLOG 441, 843
PROMON 141, 260–261, 267, 271, 463, 517, 544, 627, 863
PROQUIET 102, 104
PROREST 100, 112–113, 141, 844
PROSERVE 77
PROSTRCT 141, 326, 332, 786, 795
PROUTIL 259–261, 412, 419, 427, 436, 632
RFUTIL 142, 258, 798
SQLDUMP 849
SQLLOAD 853
SQLSCHEMA 857

V

Variable-length extents
on-line backups 93
VIEWB2 qualifier
PROUTIL utility 778
Virtual System Table 864
Virtual system tables (VSTs)
_UserStatus 344, 346, 712, 722, 726, 734, 770
VST 864
VSTs, See Virtual system tables (VSTs)

W

Watchdog process, See PROWDOG command
WBREAK-COMPILER qualifier
PROUTIL utility 781
WebSpeed
Messenger 830
Transaction Server 830
Wide area networks
connecting 490
Windows Application Event Log 453, 500
WORD-RULES qualifier
PROUTIL utility 781

Y

-yy startup parameter 480, 540, 647

Z

ZEROSTATS qualifier
PROUTIL utility 782