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.

Third party acknowledgements — See the “Third party acknowledgements” section on page Preface–10.

December 2009

Last updated with new content: Release 10.2B

Product Code: 4496; R10.2B

For the latest documentation updates see OpenEdge Product Documentation on PSDN (http://communities.progress.com/pcom/docs/DOC-16074).
## Contents

Preface .............................................. Preface–1

Part I Database Basics

1. Creating and Deleting Databases 1–1
   Ways to create an OpenEdge database 1–2
   Creating a database with PROSTRCT CREATE 1–3
      Creating a structure description file 1–3
      Create a database structure extent 1–10
      Adding schema to a void database 1–12
   Creating a database with the PRODB utility 1–13
   Creating a database with a Data tool 1–15
   Using an OpenEdge database as an Actional repository 1–16
   Converting a Progress Version 9 database to OpenEdge Release 10 1–17
      Using the Schema Mover after conversion 1–18
   Migrating from Release 10.1A 1–20
      Database migration and interoperability 1–20
      RDBMS client connections from prior releases 1–22
      Reverting your database to Release 10.1A 1–22
   Database conversion utilities 1–25
   Copying a database 1–26
      Copying a database using PROCOPY 1–26
   Deleting a database 1–28

2. OpenEdge RDBMS Limits 2–1
   Database block sizes 2–2
   Maximum number of records per block 2–3
   Number and size of storage areas 2–4
      Recovery sizes 2–9
   Table and index limits 2–10
   Number and size of sequences 2–11
   Maximum size of the primary recovery (BI) area 2–12
   Maximum database size 2–13
3. Starting Up and Shutting Down ............................................. 3–1
   Management tools ......................................................... 3–2
      OpenEdge Management ............................................... 3–2
      OpenEdge Explorer .................................................. 3–2
      Progress Explorer .................................................... 3–2
   Starting a server or broker .............................................. 3–5
      Specifying international character sets. .......................... 3–5
      Network addressing using -S and -H ............................... 3–6
      Starting multiple brokers using the same protocol .............. 3–6
      Accessing a server behind a firewall .............................. 3–8
   Starting database brokers for SSL connections ....................... 3–8
   Starting and stopping background writers ............................. 3–10
      Starting and stopping an APW ...................................... 3–10
      Starting and stopping a BIW ........................................ 3–11
      Starting and stopping an AIW ...................................... 3–11
   Stopping a server or broker ............................................ 3–12
      PROSHUT command .................................................... 3–12
      PROMON shut down database option ............................... 3–16

Part II Protecting Your Data

4. Backup Strategies .......................................................... 4–1
   Identifying files for backup .............................................. 4–2
   Determining the type of backup ........................................ 4–3
      Full backups ............................................................ 4–4
      Incremental backups ................................................... 4–4
      Online backups .......................................................... 4–6
      Offline backups .......................................................... 4–6
   Choosing backup media ................................................... 4–7
   Creating a backup schedule .............................................. 4–8
      Database integrity ....................................................... 4–8
      Database size ............................................................. 4–8
      Time ......................................................................... 4–9
      Unscheduled backups .................................................. 4–9

5. Backing Up a Database ...................................................... 5–1
   Using PROBKUP ............................................................. 5–2
      Performing an online full backup with PROBKUP ................. 5–2
      Testing backups .......................................................... 5–4
      Archiving backups ...................................................... 5–5
      Performing an offline backup ......................................... 5–6
      Performing an online backup ......................................... 5–8
   Using database quiet points ............................................. 5–9
   Performing an operating system backup ............................... 5–11
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>5. Database backup examples</strong></td>
<td>5–13</td>
</tr>
<tr>
<td>Incremental backup example</td>
<td>5–13</td>
</tr>
<tr>
<td>Full backup example</td>
<td>5–15</td>
</tr>
<tr>
<td><strong>Verifying a backup</strong></td>
<td>5–18</td>
</tr>
<tr>
<td><strong>CRC codes and redundancy in backup recovery</strong></td>
<td>5–19</td>
</tr>
<tr>
<td>CRC codes</td>
<td>5–19</td>
</tr>
<tr>
<td>Error-correction blocks</td>
<td>5–19</td>
</tr>
<tr>
<td><strong>Restoring a database</strong></td>
<td>5–20</td>
</tr>
<tr>
<td>Using the PROREST utility to restore a database</td>
<td>5–20</td>
</tr>
<tr>
<td>Important rules for restoring backups</td>
<td>5–21</td>
</tr>
<tr>
<td>Obtaining storage area descriptions using PROREST</td>
<td>5–22</td>
</tr>
<tr>
<td>Database restore examples</td>
<td>5–22</td>
</tr>
</tbody>
</table>

### 6. Recovering a Database

6–1

Introduction to recovery mechanisms | 6–2
Crash recovery | 6–3
Roll-forward recovery | 6–4
Two-phase commit | 6–5

File locations that ensure safe recovery | 6–6
Developing a recovery plan | 6–7
Time needed for recovery | 6–7
Recovery guidelines | 6–8

Sample recovery plans | 6–9
Example 1: Low availability requirements | 6–9
Example 2: Moderate availability requirements | 6–9
Example 3: Moderate-to-high availability requirements | 6–10
Example 4: High availability requirements | 6–11
Sample recovery scenarios | 6–12

After-imaging and roll-forward recovery commands | 6–17
Recovering from system failures | 6–18
System crash while running RFUTIL ROLL FORWARD | 6–18
System crash while running other utilities | 6–18
System crash while backing up the database | 6–19
System crash while database is up | 6–19

Recovering from media failures | 6–20
Loss of the DB files, BI files, or both | 6–20
Loss of the AI file | 6–21
Loss of database backup | 6–21
Loss of transaction log file | 6–22

Recovering from a full disk | 6–23
After-image area disk | 6–23
Control or primary recovery area disk | 6–23
Transaction log file disk | 6–24

Truncating the BI file | 6–25
Releasing shared memory | 6–26
Recovering from a lost or damaged control area | 6–27
Unlocking damaged databases | 6–28
Dumping tables from a damaged database | 6–29
Forcing access to a damaged database | 6–30

### 7. After-imaging

7–1

After-image areas and extents | 7–2
Estimating after-imaging disk space requirements | 7–4
Creating after-image areas | 7–5
### Contents

**8. Maintaining Security** ................................................. 8–1

- Establishing an OpenEdge user ID and password .................. 8–2
  - OpenEdge user ID .............................................. 8–2
  - OpenEdge password ........................................... 8–2
  - Validating an OpenEdge user ID and password .................. 8–3
- Establishing authentication for your OpenEdge database ........ 8–4
  - ABL tables only ............................................... 8–4
  - SQL tables only ............................................... 8–4
  - Both ABL and SQL tables ....................................... 8–4
- Connection security .................................................. 8–5
  - Designating valid users ........................................ 8–5
  - Designating a security administrator ............................ 8–6
  - Deleting a user .................................................. 8–7
  - Changing a password .......................................... 8–8
- Running a user report .............................................. 8–9
- Schema security ..................................................... 8–10
- Operating systems and database security ......................... 8–11

**9. Auditing** .......................................................... 9–1

- Auditable events ..................................................... 9–2
  - Audit events ..................................................... 9–2
  - Security events .................................................. 9–3
  - Schema events ................................................... 9–3
  - Data events ...................................................... 9–3

- Enabling after-imaging offline .................................. 7–7
  - Enabling encrypted after-imaging offline for an
    encryption-enabled database .................................. 7–7
  - Enabling unencrypted after-imaging offline for an
    encryption-enabled database .................................. 7–7
- Enabling after-imaging online .................................. 7–8
  - Enabling encrypted after-imaging online for an
    encryption-enabled database .................................. 7–8
  - Enabling unencrypted after-imaging online for an
    encryption-enabled database .................................. 7–8
- Managing after-imaging files .................................... 7–9
  - Monitoring AI file status ....................................... 7–9
  - Switching to a new AI file ..................................... 7–11
  - Archiving an AI file ............................................ 7–13
  - Making an AI file available for reuse ......................... 7–15
- AI File Management utility ....................................... 7–16
  - Automatic extent archiving .................................... 7–16
  - Enabling your database for automated AI file management .. 7–19
  - Monitoring and adjusting automated AI File Management .... 7–20
  - Restarting the AI file management utility daemon .......... 7–21
  - Archived extents ............................................... 7–21
  - Archive log file ................................................ 7–22
  - Database recovery with an archival log ....................... 7–26
- Add and reorder AI extents ....................................... 7–28
- Performing roll-forward recovery ................................ 7–30
  - Roll forward with the OPLOCK qualifier ..................... 7–31
- After-image sequences ............................................ 7–33
  - Sequence not required ........................................ 7–34
  - Sequence required ............................................. 7–35
- Disabling after-imaging ......................................... 7–36
  - Disabling encryption of AI files when disabling after-imaging 7–36

**9. Auditing** .......................................................... 9–1

- Auditable events ..................................................... 9–2
  - Audit events ..................................................... 9–2
  - Security events .................................................. 9–3
  - Schema events ................................................... 9–3
  - Data events ...................................................... 9–3
Administration events .................................................. 9–4
Utility events ............................................................. 9–4
User events ............................................................... 9–4
Application events ....................................................... 9–5
Auditing states ........................................................... 9–6
Enabling and disabling auditing ..................................... 9–7
Enabling auditing ......................................................... 9–7
Disabling auditing ......................................................... 9–8
Auditing tables .......................................................... 9–10
Indexes on auditing tables .......................................... 9–11
Archiving audit data .................................................... 9–13
Audit archive process .................................................. 9–13
Auditing impact on database resources ......................... 9–15
Auditing impact on database utilities ......................... 9–16
Identifying the privileged user ........................................ 9–16
Utility modifications .................................................... 9–19

10. Transparent Data Encryption ...................................... 10–1
Overview ........................................................................ 10–2
Before you start .......................................................... 10–2
What is encryptable ....................................................... 10–3
Data considerations ....................................................... 10–4
Database key store ......................................................... 10–4
OpenEdge supported ciphers ........................................ 10–6
Enabling Encryption ...................................................... 10–7
Adding an encryption policy area .................................... 10–7
Running ENABLEENCRYPTION ........................................ 10–8
Enabling BI file encryption after enabling encryption .... 10–9
Enabling AI file encryption after enabling encryption .... 10–10
Verifying enablements ................................................... 10–10
Creating encryption policies ......................................... 10–12
Creating encryption policies with PROUTIL EPOLICY .... 10–12
Creating encryption policies with Data Admin .............. 10–13
Encrypting your existing data ......................................... 10–14
EPOLICY MANAGE UPDATE EXAMPLE ......................... 10–14
All new data is encrypted .............................................. 10–15
Running with Transparent Data Encryption enabled .... 10–16
Manual start ............................................................... 10–16
Autostart ................................................................. 10–17
Temporary files ............................................................ 10–18
Restricted utility access ............................................... 10–18
Database backups ......................................................... 10–19
Moving tables and indexes ............................................ 10–19
Binary dump and load of data ....................................... 10–22
Auditing Transparent Data Encryption ......................... 10–22
Replication support ....................................................... 10–26
Maintaining Transparent Data Encryption ..................... 10–27
Rekey encryption policies with Data Admin ................. 10–27
Rekey encryption policies with PROUTIL EPOLICY .... 10–28
Viewing encryption policy history in Data Admin ........... 10–29
Disabling encryption ................................................... 10–30
OpenEdge SQL support for Transparent Data Encryption ........................................................................ 10–32
11. Replicating Data ......................................................... 11–1
   Replication schemes ............................................... 11–2
      Trigger-based replication ...................................... 11–2
      Log-based site replication .................................. 11–2
   Replication models .................................................. 11–2
   Database ownership models ...................................... 11–3
      Distribution model ............................................. 11–3
      Consolidation model ......................................... 11–3
   Implementing log-based site replication ...................... 11–6
      Log-based replication procedure ......................... 11–6

12. Failover Clusters ..................................................... 12–1
   Overview ............................................................... 12–2
      Related software and hardware ............................. 12–2
      Installation ....................................................... 12–3
      Configuration .................................................... 12–4
      Security ........................................................... 12–5
      Performance ....................................................... 12–5
      Logging ............................................................. 12–5
      Required cluster components ............................... 12–6
   Terms and concepts ................................................. 12–9
      Resources and dependencies ................................. 12–9
      Failure and recovery action ................................. 12–9
      Fail over policies ............................................. 12–9
   Using the PROCLUSTER command-line interface ............ 12–11
      Cluster-enabling a database .................................. 12–11
      Disabling a cluster-enabled database ...................... 12–12
      Starting a cluster-enabled database ....................... 12–13
      Stopping a cluster-enabled database ....................... 12–13
      Terminating a cluster-enabled database ................... 12–14
      Isalive and looksalive ......................................... 12–14
   Results of enabling an OpenEdge database for fail over ... 12–15
      Database UUID file (HPUX 32 and 64 bit only) ........... 12–15
      Changing the structure of the database .................... 12–15
      Adding extents on a volume group or file system different from the database (AIX only) ........... 12–15
   Platform-specific considerations ................................ 12–17
      Adding nodes where the database can be run for AIX ...... 12–17
      Upper limit on the number of packages for HPUX 32 bit and 64 bit ........................................... 12–17
      Directory of registered packages for HPUX 32 bit and 64 bit .................................................. 12–18
   Using a cluster-enabled database with the AdminServer .................................................. 12–19
   Using a cluster-enabled database with standard commands .................................................. 12–21
   Using the Windows Cluster Administrator ...................... 12–22
   Emergency disabling of a cluster-enabled database ......... 12–24
   UNIX cluster management commands .......................... 12–25

13. Distributed Transaction Processing .............................. 13–1
   Distributed transactions ............................................. 13–2
   Two-phase commit with ABL clients ............................ 13–3
      How the database engine implements two-phase commit .................. 13–3
      Two-phase commit and roll-forward recovery ............... 13–6
      Enabling two-phase commit ................................... 13–6
      Modifying the database nickname and priority ............. 13–7
      Transaction log area ............................................. 13–8
      Deactivating two-phase commit ................................ 13–8
Part III Maintaining and Monitoring Your Database

14. Managing Performance .................................................. 14–1
   Introduction to performance management ........................... 14–1
   Tools for monitoring performance ..................................... 14–2
      PROMON utility ......................................................... 14–2
      Virtual system tables ............................................... 14–3
      Windows Performance tool .......................................... 14–3
   Server performance factors ............................................ 14–4
      CPU usage ............................................................. 14–4
      Disk I/O .................................................................. 14–5
      Database I/O .......................................................... 14–5
      Before-image I/O ...................................................... 14–15
      After-image I/O ....................................................... 14–22
      Direct I/O ................................................................ 14–25
   Memory usage .................................................................. 14–26
      Process address space overview ..................................... 14–26
      Controlling memory use .............................................. 14–27
      Shared memory allocation ............................................ 14–27
      Monitoring memory use .............................................. 14–28
      Shared memory consumption and allocation impact .......... 14–28
      Shared memory summary ............................................ 14–30
   Alternate Buffer Pool ........................................................ 14–31
      Overview ................................................................. 14–31
      Designating objects to the Alternate Buffer Pool ............... 14–32
      Viewing Alternate Buffer Pool selections ......................... 14–33
      Allocating Alternate Buffer Pool ................................... 14–35
   Operating system resources .............................................. 14–36
      Processes ................................................................. 14–36
      Semaphores ............................................................. 14–36
      Spin locks .................................................................. 14–39
      File descriptors ......................................................... 14–39
   Database fragmentation .................................................... 14–40
      Analyzing database fragmentation .................................. 14–40
      Eliminating database fragmentation ............................... 14–41
      Managing fragmentation .............................................. 14–41
   Index use ....................................................................... 14–44
      Analyzing index use .................................................... 14–44
      Compacting indexes ................................................... 14–45
      Rebuilding indexes .................................................... 14–46
      Activating a single index ............................................. 14–52
   Virtual system tables ........................................................ 14–53

Java Transaction API (JTA) support ..................................... 13–18
   JTA resource impact ...................................................... 13–18
   JTA processing impact ................................................... 13–19
   Enabling JTA support ..................................................... 13–19
   Disabling JTA support .................................................... 13–19
   Monitoring JTA transactions ........................................... 13–20
   Resolving JTA transactions ............................................. 13–20

Limbo transactions with two-phase commit ............................ 13–9
Resolving limbo transactions .............................................. 13–10
Resolving limbo transaction scenarios .................................. 13–14
Two-phase commit case study ............................................ 13–15

Index use ....................................................................... 14–44
Compacting indexes ......................................................... 14–45
Rebuilding indexes .......................................................... 14–46
Activating a single index .................................................... 14–52
Virtual system tables ........................................................ 14–53
15. **Maintaining Database Structure** ........................................ 15–1
   - OpenEdge Structure utility ........................................ 15–2
   - OpenEdge Structure Statistics utility .......................... 15–3
   - OpenEdge Structure List utility ................................ 15–6
   - OpenEdge Structure Add utility .................................. 15–8
     - PROSTRCT ADD and pathnames .............................. 15–9
     - Adding extents to existing storage areas .................. 15–10
   - OpenEdge Structure Add Online utility ....................... 15–11
   - Area numbers ....................................................... 15–15
     - Area number assignment .................................... 15–15
     - Trimming unused area memory ............................... 15–19
     - Error checking .................................................. 15–21
   - Validating structure description files ......................... 15–22
   - OpenEdge Structure Remove utility ............................. 15–24
   - Maintaining indexes and tables ................................. 15–25
     - Moving tables .................................................. 15–25
     - Moving indexes ............................................... 15–26
     - Compacting indexes ........................................... 15–27
   - Performing chain analysis online ............................... 15–28
   - Using virtual system tables ..................................... 15–30

16. **Dumping and Loading** .............................................. 16–1
   - Overview of dumping and loading ............................ 16–2
     - Dump and load limitations .................................. 16–2
   - Dumping ABL database definitions .............................. 16–3
     - Definition file format ........................................ 16–5
     - Creating an incremental ABL data definitions file ........ 16–6
     - Dumping sequence definitions ............................... 16–7
     - Dumping auto-connect records ............................... 16–8
   - Dumping database contents ..................................... 16–9
     - Dumping table contents with PROUTIL ...................... 16–9
     - Dumping selective contents with PROUTIL ................. 16–11
     - Dumping table contents with a Data tool ................. 16–13
     - Contents file format ......................................... 16–14
     - Dumping sequence values with a Data tool ............... 16–16
     - Dumping user table contents with a Data tool .......... 16–16
     - Dumping an SQL view file’s contents with a Data tool . 16–17
   - Loading database definitions ................................... 16–18
     - Loading updated ABL data definitions ................. 16–18
   - Loading database contents ..................................... 16–20
     - Loading table contents in binary format with PROUTIL . 16–20
     - Loading table contents with a Data tool .................. 16–22
     - Loading user table contents with a Data tool ........... 16–23
     - Loading an SQL view file contents ......................... 16–23
     - Loading sequence values using a Data tool ................ 16–24
   - Bulk loading ....................................................... 16–25
     - Creating a Bulk Loader description file .................. 16–25
     - Modifying a Bulk Loader description file ............... 16–26
     - Loading table contents with the Bulk Loader qualifier . 16–27
   - Reconstructing bad load records ............................. 16–29
   - Specialized dump and load techniques ....................... 16–30
     - Creating a starting version of a database ............... 16–30
     - Using a constant table in multiple databases .......... 16–31
     - Economizing disk space ...................................... 16–31
## Contents

Optimizing data for sequential access ............................................. 16–32  
Optimizing data for random access ................................................... 16–33  
Using no-integrity mode bulk data loading ........................................ 16–34  
Saving disk space by deactivating indexes ....................................... 16–34

### 17. Logged Data ........................................................................ 17–1

- OpenEdge Release 10 database log file ............................................. 17–2
- Managing log file size .................................................................. 17–4
- Saving key database events ............................................................. 17–5
  - Defining key events ................................................................. 17–5
  - Saving key events ................................................................... 17–5
  - Enabling save key events .......................................................... 17–6
  - Disabling save key events ........................................................ 17–7
  - Stored key events ................................................................... 17–7
  - _KeyEvt table ....................................................................... 17–11
- Event logging in Windows ............................................................. 17–13
  - Managing OpenEdge RDBMS events in Windows ....................... 17–13
  - Understanding the event log components ................................... 17–14
  - The Event Log and the registry ................................................. 17–15
- Client database-request statement caching ..................................... 17–16

### Part IV Reference

### 18. Startup and Shutdown Commands ......................................... 18–1

- Startup command syntax ............................................................. 18–2
- Database startup and shutdown commands .................................... 18–3
  - PROAIW command .................................................................. 18–4
  - PROAPW command .................................................................. 18–5
  - PROBIW command ................................................................... 18–6
  - PROQUIET command ................................................................ 18–7
  - PROSERVE command ................................................................ 18–9
  - PROSHUT command .................................................................. 18–11
  - PROWDOG command ................................................................ 18–15

### 19. Database Startup Parameters ............................................... 19–1

- Issuing startup parameters ........................................................... 19–2
- Database server performance parameters ...................................... 19–3
- Database server-type parameters .................................................. 19–5
- Database server internationalization parameters ............................ 19–6
- Database server statistics parameters .......................................... 19–7
- Database server consistency check parameters ......................... 19–8
- Database server SSL connection parameters ............................... 19–9
- Database server network parameters ........................................... 19–10
- Startup parameter usage categories .............................................. 19–12
- Alphabetical listing of database startup parameters .................... 19–13
  - AdminServer Port (-adminport) ............................................... 19–13
  - After-image File Management Archive Directory List (-aiarcdir) .... 19–14
  - After-image File Management Archive Directory Create (-aiarcdircreate) 19–14
  - After-image File Management Archive Interval (-aiarcinterval) ........ 19–15
  - After-image Buffers (-aibufs) .................................................. 19–16
  - After-image Stall (-aistall) ...................................................... 19–16
  - Area consistency check (-AreaCheck) ......................................... 19–17
  - Blocks in Database Buffers (-B) ................................................. 19–18
  - Blocks in Alternate Buffer Pool (-B2) ......................................... 19–19
Contents–10

Base Index (-baseindex) .................................................. 19–20
Base Table (-basetable) ............................................... 19–21
Before-image Buffers (-bibufs) ....................................... 19–22
Threshold Stall (-bistall) .............................................. 19–22
Recovery Log Threshold (-bithold) .................................. 19–23
Schema Cache File (-cache) ........................................... 19–24
SQL Server Java Classpath (-classpath) ......................... 19–25
Cluster Mode (-cluster) ................................................. 19–26
Conversion Map (-convmap) .......................................... 19–27
Case Table (-cpcase) .................................................... 19–27
Collation Table (-cpcoll) .............................................. 19–28
Internal Code Page (-cpprint) ....................................... 19–29
Log File Code Page (-cplog) .......................................... 19–29
Print Code Page (-cprint) ............................................. 19–30
R-code in Code Page (-cprcodein) ................................. 19–31
Stream Code Page (-cpstream) ....................................... 19–32
Terminal Code Page (-cpterm) ....................................... 19–33
Database consistency check (-DbCheck) ......................... 19–34
Direct I/O (-directio) .................................................... 19–35
Encryption cache size (-ecsize) ....................................... 19–35
Event Level (-evtlevel) .................................................. 19–36
Before-image Truncate Interval (-G) ............................... 19–37
Group Delay (-groupdelay) ............................................ 19–37
Host Name (-H) ........................................................... 19–38
Hash Table Entries (-hash) ........................................... 19–38
No Crash Protection (-i) .............................................. 19–39
Index consistency check (-IndexCheck) ......................... 19–40
Index Range Size (-indexrangesize) .............................. 19–41
Internet Protocol (-ipver) ............................................ 19–41
Key Alias (-keyalias) .................................................... 19–42
Key Alias Password (-keyaliaspasswd) ....................... 19–43
Lock Table Entries (-L) ................................................. 19–43
Lock Table Hash Size (-lkhash) ..................................... 19–44
Lock release (-lkrela) .................................................. 19–44
Auto Server (-m1) ........................................................ 19–45
Manual Server (-m2) ..................................................... 19–45
Secondary Login Broker (-m3) ..................................... 19–45
Maximum Clients per Server (-Ma) .................... 19–46
Maximum area number (-maxAreas) ............................. 19–46
Maximum Dynamic Server (-maxdynserver) .................... 19–47
Maximum JTA Transactions (-maxtx) ......................... 19–47
Memory consistency check (-MemCheck) ....................... 19–48
Delayed BI File Write (-Mf) ......................................... 19–49
Minimum Clients per Server (-mi) ............................... 19–50
Minimum Dynamic Server (-minport) ....................... 19–50
Maximum Servers (-Mn) ................................................ 19–51
Servers per Protocol (-Mp) ......................................... 19–51
Maximum Servers per Broker (-Mbp) ....................... 19–52
VLM Page Table Entry Optimization (-Mpte) ................... 19–52
User MUX Latches (-muix) .......................................... 19–52
Shared-memory Overflow Size (-Mxs) ......................... 19–53
Network Type (-N) ....................................................... 19–54
Number of Users (-n) .................................................. 19–54
Nap Time Increment (-napinc) ...................................... 19–55
Nap Time Steps (-napstep) ............................................ 19–55
No Session Cache (-nosessioncache) ......................... 19–55
Storage Object Cache Size (-omsize) ....................... 19–56
<table>
<thead>
<tr>
<th>20. PROMON Utility</th>
<th>20–1</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROMON utility</td>
<td>20–2</td>
</tr>
<tr>
<td>PROMON User Control option</td>
<td>20–4</td>
</tr>
<tr>
<td>PROMON Locking and Waiting statistics option</td>
<td>20–9</td>
</tr>
<tr>
<td>PROMON Block Access option</td>
<td>20–11</td>
</tr>
<tr>
<td>PROMON Record Locking Table option</td>
<td>20–13</td>
</tr>
<tr>
<td>PROMON Activity option</td>
<td>20–16</td>
</tr>
<tr>
<td>PROMON Shared Resources option</td>
<td>20–19</td>
</tr>
<tr>
<td>PROMON Database Status option</td>
<td>20–21</td>
</tr>
<tr>
<td>PROMON Shut Down Database option</td>
<td>20–23</td>
</tr>
<tr>
<td>PROMON R&amp;D Advanced Options</td>
<td>20–24</td>
</tr>
<tr>
<td>R&amp;D Status Displays</td>
<td>20–25</td>
</tr>
<tr>
<td>Database</td>
<td>20–26</td>
</tr>
<tr>
<td>Backup</td>
<td>20–28</td>
</tr>
<tr>
<td>Servers</td>
<td>20–29</td>
</tr>
<tr>
<td>Processes/Clients</td>
<td>20–30</td>
</tr>
<tr>
<td>Files</td>
<td>20–34</td>
</tr>
<tr>
<td>Lock Table</td>
<td>20–35</td>
</tr>
<tr>
<td>Buffer Cache</td>
<td>20–37</td>
</tr>
<tr>
<td>Logging Summary</td>
<td>20–38</td>
</tr>
<tr>
<td>BI Log</td>
<td>20–39</td>
</tr>
<tr>
<td>AI Log</td>
<td>20–40</td>
</tr>
<tr>
<td>Two-phase Commit</td>
<td>20–41</td>
</tr>
<tr>
<td>Startup Parameters</td>
<td>20–42</td>
</tr>
<tr>
<td>Shared Resources</td>
<td>20–43</td>
</tr>
<tr>
<td>Shared Memory Segments</td>
<td>20–44</td>
</tr>
<tr>
<td>AI Extents</td>
<td>20–45</td>
</tr>
<tr>
<td>Database Service Manager</td>
<td>20–46</td>
</tr>
<tr>
<td>Servers by broker</td>
<td>20–48</td>
</tr>
<tr>
<td>Client Database-Request Statement Cache</td>
<td>20–49</td>
</tr>
<tr>
<td>R&amp;D Activity Displays</td>
<td>20–57</td>
</tr>
<tr>
<td>Summary</td>
<td>20–59</td>
</tr>
<tr>
<td>Servers</td>
<td>20–62</td>
</tr>
<tr>
<td>Buffer Cache</td>
<td>20–63</td>
</tr>
<tr>
<td>Page Writers</td>
<td>20–65</td>
</tr>
<tr>
<td>BI Log</td>
<td>20–66</td>
</tr>
<tr>
<td>AI Log</td>
<td>20–68</td>
</tr>
<tr>
<td>Lock Table</td>
<td>20–69</td>
</tr>
</tbody>
</table>

Prompt for Passphrase (-Passphrase) .......................... 19–56
Pending Connection Time (-PendConnTime) .......................... 19–57
Parameter File (-pf) .................................................. 19–57
Database Service Communication Area Size (-pica) ............. 19–58
Pin Shared Memory (-pinshm) ........................................ 19–58
Configuration Properties File (-properties) ...................... 19–58
Buffered I/O (-r) ...................................................... 19–59
Service Name (-S) ....................................................... 19–59
Semaphore Sets (-semsets) ........................................... 19–60
Server Group (-servergroup). ....................................... 19–61
Session Timeout (-sessiontimeout) .................................. 19–61
Shared memory segment size (-shmsegsize) ....................... 19–62
Spin Lock Retries (-spin) ............................................ 19–63
SSL (-ssl) .............................................................. 19–64
Table consistency check (-TableCheck) .......................... 19–64
Table Range Size (-tablerangesize) ............................... 19–65
Century Year Offset (-yy) ........................................... 19–66
21. PROUTIL Utility ................................................. 21–1
   PROUTIL utility syntax .................................. 21–2
   PROUTIL 2PHASE BEGIN qualifier ..................... 21–7
   PROUTIL 2PHASE COMMIT qualifier ..................... 21–8
   PROUTIL 2PHASE END qualifier ......................... 21–9
   PROUTIL 2PHASE MODIFY qualifier ..................... 21–10
   PROUTIL 2PHASE RECOVER qualifier .................... 21–11
   PROUTIL AUDITARCHIVE qualifier ....................... 21–12
   PROUTIL AUDITLOAD qualifier ............................ 21–15
   PROUTIL BIGROW qualifier ............................... 21–17
   PROUTIL BULKLOAD qualifier ............................. 21–18
   PROUTIL BUSY qualifier .................................. 21–19
   PROUTIL CHANALYS qualifier ............................. 21–20
   PROUTIL CODEPAGE-COMPILER qualifier ................. 21–21
   PROUTIL CONV910 qualifier ............................. 21–22
   PROUTIL CONVCHAR qualifier ............................ 21–23
   PROUTIL CONVFIELE qualifier ........................... 21–26
   PROUTIL DBANALYS qualifier ............................. 21–28
   PROUTIL DBAUTHKEY qualifier ............................ 21–29
   PROUTIL DESCRIBE qualifier ............................. 21–30
   PROUTIL DBIPCS qualifier ................................ 21–34
   PROUTIL DISABLEAUDITING qualifier .................... 21–35
   PROUTIL DISABLEB2 ....................................... 21–36
   PROUTIL DISABLEENCRYPTION ......................... 21–37
   PROUTIL DISABLEJTA qualifier .......................... 21–39
   PROUTIL DISABLEKEYEVENTS .............................. 21–40
   PROUTIL DISPTOSSCREATELIMITS qualifier ............... 21–41
<table>
<thead>
<tr>
<th>Module</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROUTIL DUMP qualifier</td>
<td>21–42</td>
</tr>
<tr>
<td>PROUTIL DUMPSPECIFIED qualifier</td>
<td>21–45</td>
</tr>
<tr>
<td>PROUTIL ENABLEAUDITING qualifier</td>
<td>21–47</td>
</tr>
<tr>
<td>PROUTIL ENABLEB2</td>
<td>21–49</td>
</tr>
<tr>
<td>PROUTIL ENABLECRIPTON</td>
<td>21–50</td>
</tr>
<tr>
<td>PROUTIL ENABLEJTA qualifier</td>
<td>21–52</td>
</tr>
<tr>
<td>PROUTIL ENABLEKEYEVENTS qualifier</td>
<td>21–53</td>
</tr>
<tr>
<td>PROUTIL ENABLELARGEFILES qualifier</td>
<td>21–54</td>
</tr>
<tr>
<td>PROUTIL ENABLELARGEKEYS qualifier</td>
<td>21–55</td>
</tr>
<tr>
<td>PROUTIL ENABLESEQ64 qualifier</td>
<td>21–56</td>
</tr>
<tr>
<td>PROUTIL ENABLESTOREDPROC qualifier</td>
<td>21–57</td>
</tr>
<tr>
<td>PROUTIL EPOLICY INFO qualifier</td>
<td>21–58</td>
</tr>
<tr>
<td>PROUTIL EPOLICY MANAGE qualifier</td>
<td>21–59</td>
</tr>
<tr>
<td>PROUTIL EPOLICY SCAN qualifier</td>
<td>21–61</td>
</tr>
<tr>
<td>PROUTIL EPOLICY VIEW qualifier</td>
<td>21–62</td>
</tr>
<tr>
<td>PROUTIL HOLDER qualifier</td>
<td>21–63</td>
</tr>
<tr>
<td>PROUTIL IDXACTIVATE qualifier</td>
<td>21–64</td>
</tr>
<tr>
<td>PROUTIL IDXANALYS qualifier</td>
<td>21–65</td>
</tr>
<tr>
<td>PROUTIL IDXBUILD qualifier</td>
<td>21–66</td>
</tr>
<tr>
<td>PROUTIL IDXCHECK qualifier</td>
<td>21–70</td>
</tr>
<tr>
<td>PROUTIL IDXCOMPACT qualifier</td>
<td>21–73</td>
</tr>
<tr>
<td>PROUTIL IDXFIX qualifier</td>
<td>21–75</td>
</tr>
<tr>
<td>PROUTIL IDXMOVE qualifier</td>
<td>21–79</td>
</tr>
<tr>
<td>PROUTIL INCREASETO qualifier</td>
<td>21–81</td>
</tr>
<tr>
<td>PROUTIL IOSTATS qualifier</td>
<td>21–83</td>
</tr>
<tr>
<td>PROUTIL LOAD qualifier</td>
<td>21–85</td>
</tr>
<tr>
<td>PROUTIL MVSCH qualifier</td>
<td>21–88</td>
</tr>
<tr>
<td>PROUTIL RCODEKEY qualifier</td>
<td>21–89</td>
</tr>
<tr>
<td>PROUTIL REVERT qualifier</td>
<td>21–90</td>
</tr>
<tr>
<td>PROUTIL SETAREACREATELIMIT qualifier</td>
<td>21–93</td>
</tr>
<tr>
<td>PROUTIL SETAREATOSSLIMIT qualifier</td>
<td>21–94</td>
</tr>
<tr>
<td>PROUTIL SETBLOBCREATELIMIT qualifier</td>
<td>21–95</td>
</tr>
<tr>
<td>PROUTIL SETBLOBTOSSLIMIT qualifier</td>
<td>21–96</td>
</tr>
<tr>
<td>PROUTIL SETTABLECREATELIMIT qualifier</td>
<td>21–97</td>
</tr>
<tr>
<td>PROUTIL SETTABLETOSSLIMIT qualifier</td>
<td>21–98</td>
</tr>
<tr>
<td>PROUTIL TABANALYS qualifier</td>
<td>21–99</td>
</tr>
<tr>
<td>PROUTIL TABLEMOVE qualifier</td>
<td>21–101</td>
</tr>
<tr>
<td>PROUTIL TRUNCATE AREA qualifier</td>
<td>21–103</td>
</tr>
<tr>
<td>PROUTIL TRUNCATE BI qualifier</td>
<td>21–105</td>
</tr>
<tr>
<td>PROUTIL UPDATESCHEMA qualifier</td>
<td>21–107</td>
</tr>
<tr>
<td>PROUTIL UPDATEVST qualifier</td>
<td>21–108</td>
</tr>
<tr>
<td>PROUTIL VIEWB2</td>
<td>21–109</td>
</tr>
<tr>
<td>PROUTIL WBREAK-COMPILER qualifier</td>
<td>21–115</td>
</tr>
</tbody>
</table>

22. **PROSTRCT Utility** .......................... 22-1

PROSTRCT utility ................................ 22-2
PROSTRCT ADD qualifier .......................... 22-4
PROSTRCT ADDONLINE qualifier .................... 22-5
PROSTRCT BUILDDB qualifier ....................... 22-7
PROSTRCT CLUSTER CLEAR qualifier ................ 22-8
PROSTRCT CREATE qualifier ........................ 22-9
PROSTRCT LIST qualifier ........................... 22-11
PROSTRCT REMOVE qualifier ........................ 22-12
## PROSTRUCT Qualifiers

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>REORDER AI</td>
<td>22–13</td>
</tr>
<tr>
<td>REPAIR</td>
<td>22–14</td>
</tr>
<tr>
<td>STATISTICS</td>
<td>22–15</td>
</tr>
<tr>
<td>UNLOCK</td>
<td>22–16</td>
</tr>
</tbody>
</table>

## RFUTIL Utility

<table>
<thead>
<tr>
<th>Syntax</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>utility syntax</td>
<td>23–1</td>
</tr>
<tr>
<td>AiArchiver Disable</td>
<td>23–5</td>
</tr>
<tr>
<td>AiArchiver Enable</td>
<td>23–6</td>
</tr>
<tr>
<td>AiArchiver End</td>
<td>23–7</td>
</tr>
<tr>
<td>AiArchiver SetDir</td>
<td>23–8</td>
</tr>
<tr>
<td>AiArchiver SetInterval</td>
<td>23–9</td>
</tr>
<tr>
<td>Archive Extent</td>
<td>23–10</td>
</tr>
<tr>
<td>Aimage Aioff</td>
<td>23–11</td>
</tr>
<tr>
<td>Aimage Begin</td>
<td>23–12</td>
</tr>
<tr>
<td>Aimage End</td>
<td>23–13</td>
</tr>
<tr>
<td>Aimage Extent Empty</td>
<td>23–14</td>
</tr>
<tr>
<td>Aimage Extent Full</td>
<td>23–15</td>
</tr>
<tr>
<td>Aimage Extent List</td>
<td>23–16</td>
</tr>
<tr>
<td>Aimage Extract</td>
<td>23–17</td>
</tr>
<tr>
<td>Aimage New</td>
<td>23–18</td>
</tr>
<tr>
<td>Aimage Query</td>
<td>23–19</td>
</tr>
<tr>
<td>Aimage Scan</td>
<td>23–21</td>
</tr>
<tr>
<td>Aimage Truncate</td>
<td>23–22</td>
</tr>
<tr>
<td>Aimage Verify Partial</td>
<td>23–23</td>
</tr>
<tr>
<td>Aimage Verify Full</td>
<td>23–24</td>
</tr>
<tr>
<td>Mark Backedup</td>
<td>23–25</td>
</tr>
<tr>
<td>Roll Forward</td>
<td>23–26</td>
</tr>
<tr>
<td>Roll Forward Oplock</td>
<td>23–28</td>
</tr>
<tr>
<td>Roll Forward Retry</td>
<td>23–29</td>
</tr>
<tr>
<td>Roll Opunlock</td>
<td>23–31</td>
</tr>
<tr>
<td>Sequence</td>
<td>23–32</td>
</tr>
</tbody>
</table>

## Other Database Administration Utilities

<table>
<thead>
<tr>
<th>Utility</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBMAN</td>
<td>24–3</td>
</tr>
<tr>
<td>DBTOOL</td>
<td>24–7</td>
</tr>
<tr>
<td>PROADSV</td>
<td>24–11</td>
</tr>
<tr>
<td>PROBKUP</td>
<td>24–13</td>
</tr>
<tr>
<td>PROCLUSTER</td>
<td>24–17</td>
</tr>
<tr>
<td>PROCOPY</td>
<td>24–21</td>
</tr>
<tr>
<td>PRODB</td>
<td>24–23</td>
</tr>
<tr>
<td>PRODEL</td>
<td>24–25</td>
</tr>
<tr>
<td>Progress Explorer</td>
<td>24–26</td>
</tr>
<tr>
<td>PROLOG</td>
<td>24–27</td>
</tr>
<tr>
<td>PROREST</td>
<td>24–28</td>
</tr>
</tbody>
</table>

## SQL Utilities

<table>
<thead>
<tr>
<th>Utility</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQLDUMP</td>
<td>25–2</td>
</tr>
<tr>
<td>SQLLOAD</td>
<td>25–6</td>
</tr>
<tr>
<td>SQLSCHEMA</td>
<td>25–10</td>
</tr>
</tbody>
</table>
26. Virtual System Tables .......................................................... 26–1
   Update access to virtual system tables .................................... 26–2
   Accessing VSTs with ABL or SQL .......................................... 26–3
   Virtual system table summaries ............................................. 26–4
   OpenEdge virtual system table schema descriptions .................. 26–9

Index .................................................................................. Index–1
Figures
Figure 1–1: How the Schema Mover works .................................................. 1–19
Figure 4–1: Incremental backup—size varies ......................................... 4–4
Figure 4–2: Incremental backup with overlap ......................................... 4–5
Figure 6–1: OpenEdge recovery mechanisms ......................................... 6–2
Figure 6–2: Storing database files ......................................................... 6–6
Figure 6–3: Recovery scenario—day 1 ..................................................... 6–12
Figure 6–4: Recovery scenario—day 2 ..................................................... 6–13
Figure 6–5: Recovery scenario—day 3 ..................................................... 6–14
Figure 6–6: Recovery scenario—day 4 ..................................................... 6–15
Figure 6–7: Recovery scenario—day 5 ..................................................... 6–16
Figure 7–1: After-image extents switching ............................................. 7–2
Figure 7–2: AI extent state transitions .................................................... 7–18
Figure 7–3: Original database AI extents ................................................. 7–28
Figure 7–4: Database AI Extents after PROSTRCT ADD of two new extents .. 7–29
Figure 7–5: Database AI extents after PROSTRCT REORDER AI .......... 7–29
Figure 9–1: Audit archive process .......................................................... 9–13
Figure 10–1: Output of PROUTILITY DESCRIBE .................................. 10–11
Figure 11–1: Data distribution model ..................................................... 11–3
Figure 11–2: Data consolidation models ................................................ 11–4
Figure 11–3: Peer-to-peer model .......................................................... 11–5
Figure 11–4: Example of site replication ................................................. 11–7
Figure 12–1: Cluster component configuration ...................................... 12–6
Figure 12–2: Relationship between database commands and utilities and the cluster manager .......................................................... 12–7
Figure 12–3: Virtual network cluster components ................................... 12–8
Figure 13–1: Data inconsistency ............................................................. 13–2
Figure 13–2: Two-phase commit algorithm ............................................ 13–4
Figure 13–3: Limbo transaction .............................................................. 13–5
Figure 13–4: How PROUTILITY processes react to limbo transactions .... 13–9
Figure 14–1: Database I/O ................................................................. 14–6
Figure 14–2: Reads and writes to database buffers ............................... 14–7
Figure 14–3: Evicting buffers ............................................................... 14–8
Figure 14–4: Monitoring buffer activity ............................................... 14–9
Figure 14–5: Block Table (BLKTBL) chain .......................................... 14–12
Figure 14–6: APWs and the least recently used chain ............................ 14–13
Figure 14–7: PROMON Page Writers Activity display ......................... 14–14
Figure 14–8: PROMON BI Log Activity display .................................... 14–16
Figure 14–9: BI clusters at startup ...................................................... 14–18
Figure 14–10: BI clusters over time ..................................................... 14–19
Figure 14–11: PROMON AI Log Activity display .................................. 14–23
Figure 14–12: Sample 4GB process address space ............................... 14–26
Figure 14–13: How a semaphore coordinates concurrent access .......... 14–37
Figure 15–1: Sequential area allocation ............................................... 15–16
Figure 15–2: Area allocation with assigned area numbers .................... 15–18
Figure 15–3: Area array trimmed by -maxAreas startup parameter ....... 15–20
Figure 16–1: Sample data definitions file .......................................... 16–5
Figure 16–2: Sample contents file ...................................................... 16–14
Figure 16–3: Dictionary-created description file example ..................... 16–26
Figure 16–4: Example modified Bulk Loader description file ................. 16–27
Figure 16–5: Copying the starting version of a database ................. 16–30
Figure 20–1: PROMON utility main menu ........................................... 20–2
Figure 20–2: Sample output for PROMON User Control option .......... 20–4
Figure 20–3: Sample output for PROMON Locking and Waiting statistics option .. 20–9
Figure 20–4: Sample output for PROMON Block Access option .......... 20–11
Figure 20–5: Sample output for PROMON Record Locking Table option .. 20–13
Figure 20–61: PROMON Modify Defaults menu ........................................ 20–97
Figure 20–60: PROMON Resolve JTA Transactions menu .................. 20–96
Figure 20–59: Sample output for PROMON Coordinator Information option 20–95
Figure 20–58: Sample output for PROMON Resolve 2PC Limbo Transactions option 20–94
Figure 20–57: Sample output for PROMON Coordinator Information option 20–95
Figure 20–56: PROMON Adjust Monitor Options menu ...................... 20–91
Figure 20–55: PROMON enable/disable block level consistency check display 20–89
Figure 20–54: PROMON Terminate a Server display ............................. 20–88
Figure 20–53: PROMON Adjust Page Writer Options menu ................. 20–87
Figure 20–52: PROMON Adjust Latch Options menu ............................ 20–87
Figure 20–51: PROMON Resolve Limbo Transactions menu ................. 20–86
Figure 20–50: PROMON Check Active Transaction Status display ........ 20–85
Figure 20–49: PROMON I/O Operations by User by Index display ........ 20–83
Figure 20–48: PROMON I/O Operations by User by Table display ........ 20–82
Figure 20–47: PROMON Checkpoints display ...................................... 20–81
Figure 20–46: PROMON Lock Requests by User display ...................... 20–80
Figure 20–45: PROMON I/O Operations by Process display ................. 20–79
Figure 20–44: PROMON Performance Indicators display ...................... 20–77
Figure 20–43: PROMON Page Writers Activity display ....................... 20–65
Figure 20–42: PROMON Record Activity display ................................. 20–75
Figure 20–41: PROMON Index Activity display .................................... 20–74
Figure 20–40: PROMON Space Allocation Activity display ................. 20–73
Figure 20–39: PROMON I/O Operations by File Activity display ......... 20–72
Figure 20–38: PROMON I/O Operations by Type Activity display ........ 20–71
Figure 20–37: PROMON Lock Table Activity display ............................ 20–69
Figure 20–36: PROMON AI Log Activity display ................................... 20–68
Figure 20–35: PROMON BI Log Activity display .................................... 20–66
Figure 20–34: PROMON Page Writers Activity display ....................... 20–65
Figure 20–33: PROMON Buffer Cache Activity display ....................... 20–63
Figure 20–32: PROMON Servers Activity display ................................. 20–62
Figure 20–31: PROMON Summary Activity display .............................. 20–61
Figure 20–30: PROMON AI Extents status display ............................... 20–45
Figure 20–29: PROMON Shared Memory Segments Status display ........ 20–43
Figure 20–28: PROMON Shared Resources Status display ................... 20–42
Figure 20–27: PROMON Startup Parameters Status display ................. 20–41
Figure 20–26: PROMON Two-phase Commit Status display ................. 20–40
Figure 20–25: PROMON AI Log Status display .................................... 20–39
Figure 20–24: PROMON BI Log Status display .................................... 20–38
Figure 20–23: PROMON Logging Summary Status display ................... 20–37
Figure 20–22: PROMON Buffer Cache Status display ........................... 20–35
Figure 20–21: PROMON Lock Table Status display .............................. 20–34
Figure 20–20: PROMON Files Status display ....................................... 20–33
Figure 20–19: PROMON Background Processes Status display .............. 20–32
Figure 20–18: PROMON Remote Clients Status display ....................... 20–33
Figure 20–17: PROMON Local Batch Clients Status display ................. 20–32
Figure 20–16: PROMON Local Interactive Clients Status display .......... 20–31
Figure 20–15: PROMON Active Transactions Status display ............... 20–31
Figure 20–14: PROMON Blocked Clients Status display ...................... 20–32
Figure 20–13: PROMON Processes/Clients Status display ................... 20–31
Figure 20–12: PROMON Servers Status display ................................... 20–29
Figure 20–11: PROMON Backup Status display ................................... 20–28
Figure 20–10: PROMON Database Status display .............................. 20–26
Figure 20–9: Sample output for PROMON Shut Down Database option 20–23
Figure 20–8: Sample output for PROMON Database Status option ....... 20–21
Figure 20–7: Sample Output for PROMON Shared Resources option ...... 20–19
Figure 20–6: Sample Output for PROMON Activity option .................. 20–16
Figure 20–5: Sample output for PROMON Activity option ................... 20–15
Figure 20–4: Sample output for PROMON Activity option ................... 20–14
Figure 20–3: Sample output for PROMON Activity option ................... 20–13
Figure 20–2: Sample output for PROMON Activity option ................... 20–12
Figure 20–1: Sample output for PROMON Activityoption ................... 20–11
Tables

Table 1–1: ST file tokens ............................................. 1–5
Table 1–2: Calculating extent size .................................. 1–8
Table 2–1: Storage area types ...................................... 2–4
Table 2–2: Maximum rows per Type II storage area (approximate) ........ 2–5
Table 2–3: Type I storage area maximums ......................... 2–7
Table 2–4: RDBMS reserved recovery space ....................... 2–9
Table 2–5: Maximum number of sequences ....................... 2–11
Table 2–6: Maximum primary recovery (BI) area size ......... 2–12
Table 2–7: Maximum number of connections per database .......... 2–14
Table 2–8: Maximum number of simultaneous transactions .... 2–15
Table 2–9: Database name limits .................................. 2–16
Table 2–10: SQL data type limits .................................. 2–19
Table 2–11: ABL data type limits ................................ 2–20
Table 2–12: ABL and SQL data type correspondence .......... 2–21
Table 3–1: PROSHUT menu options .............................. 3–14
Table 3–2: PROSHUT Parameters ................................. 3–14
Table 4–1: Backup options ........................................ 4–3
Table 4–2: Backup media questions ............................... 4–7
Table 5–1: PROREST utility verification parameters ............ 5–4
Table 6–1: Sample low availability requirements ................ 6–9
Table 6–2: Sample moderate availability requirements ........ 6–9
Table 6–3: Sample moderate-to-high availability requirements . 6–10
Table 6–4: Sample high availability requirements ............... 6–11
Table 6–5: Utilities used with roll-forward recovery ............ 6–17
Table 6–6: Shared-memory segment status fields ............... 6–26
Table 7–1: File tokens .............................................. 7–5
Table 7–2: Archival log record types ............................. 7–23
Table 7–3: Format of an archival log startup line ............ 7–24
Table 7–4: Format of an AI extent archive record in the archive log file .... 7–24
Table 7–5: Format of a backup record in the archive log file ... 7–25
Table 7–6: Auditing meta-schema tables ......................... 9–10
Table 8–1: Indexes for auditing schema tables ................. 9–11
Table 8–2: Encrypted password components ..................... 9–19
Table 8–3: Protected tables ....................................... 9–20
Table 8–4: Protected utilities ..................................... 9–20
Table 10–1: Passphrase constraints .............................. 10–5
Table 10–2: Database Master key ciphers ....................... 10–6
Table 10–3: Object ciphers ...................................... 10–6
Table 10–4: Encryption impact of moving tables and indexes ... 10–19
Table 10–5: Auditing events for Transparent Data Encryption ... 10–22
Table 12–1: UNIX cluster management commands .......... 12–25
Table 12–2: UNIX Shared disk commands ....................... 12–26
Table 14–1: Startup parameters for I/O by object .......... 14–15
Table 14–2: Startup parameters that affect memory allocation .. 14–27
Table 14–3: Shared memory problems and solutions ........ 14–29
Table 14–4: UNIX kernel parameters that affect semaphores ... 14–38
Table 14–5: PROUTIL qualifiers for changing create and toss limits ... 14–42
Table 14–6: Create and toss limit situations and solutions ... 14–42
Table 14–7: Factor values ....................................... 14–45
Table 16–1: Definitions dumped to the definition file ......... 16–3
Table 16–2: Data definitions file trailer values ................ 16–6
Table 16–3: DUMPSPECIFIED syntax .......................... 16–11
Table 16–4: Contents file trailer variables ....................... 16–15
Table 16–5: Modifying the description variables ............... 16–26
Table 17–1: Database event log format ......................... 17–2
Table 26–10: Page writer activity file (_ActPWs) .................................................. 26–14
Table 26–11: Record activity file (_ActRecord) ....................................................... 26–15
Table 26–12: Server activity file (_ActServer) ......................................................... 26–15
Table 26–13: Space allocation activity file (_ActSpace) .......................................... 26–16
Table 26–14: Summary activity file (_ActSummary) ................................................ 26–17
Table 26–15: Area status file (_AreaStatus) ......................................................... 26–18
Table 26–16: Area threshold file (_AreaThreshold) ............................................... 26–18
Table 26–17: Block file (_Block) ................................................................. 26–18
Table 26–18: Buffer status file (_BuffStatus) ......................................................... 26–19
Table 26–19: Checkpoint File (_Checkpoint) ......................................................... 26–19
Table 26–20: Code Features (_Code-Feature) ......................................................... 26–20
Table 26–21: Database connection file (_Connect) .................................................. 26–20
Table 26–22: Database features (_Database-Feature) ............................................ 26–23
Table 26–23: Database status file (_DbStatus) .................................................... 26–23
Table 26–24: Database file status file (_Filelist) .................................................... 26–25
Table 26–25: Index statistics file (_IndexStat) ....................................................... 26–25
Table 26–26: Latch statistics file (_Latch) .......................................................... 26–26
Table 26–27: License management (_License) ....................................................... 26–26
Table 26–28: Lock table status file (_Lock) .......................................................... 26–27
Table 26–29: Lock request file (_LockReq) .......................................................... 26–27
Table 26–30: Logging file (_Logging) ................................................................. 26–28
Table 26–31: Master block file (_MstrBlk) ............................................................ 26–29
Table 26–32: User connection (_MyConnection) .................................................... 26–30
Table 26–33: Resource queue statistics file (_Resrc) ............................................. 26–31
Table 26–34: Segments file (_Segments) .............................................................. 26–31
Table 26–35: Servers file (_Servers) ................................................................. 26–31
Table 26–36: Startup file (_Startup) ................................................................. 26–32
Table 26–37: Index and table statistics range (_StatBase) ...................................... 26–33
Table 26–38: Table statistics file (_TableStat) ...................................................... 26–33
Table 26–39: Transaction file (_Trans) ............................................................... 26–33
Table 26–40: Transaction end lock statistics (_TxeLock) ....................................... 26–34
Table 26–41: User index activity (_UserIndexStat) .............................................. 26–34
Table 26–42: Database input/output file (_UserIO) .............................................. 26–35
Table 26–43: Record locking table file (_UserLock) ............................................ 26–35
Table 26–44: User status (_UserStatus) ............................................................ 26–36
Table 26–45: _UserStatus-State descriptions ...................................................... 26–36
Table 26–46: User table activity (_UserTableStat) ............................................... 26–39
Preface

This Preface contains the following sections:

• Purpose
• Audience
• Organization
• Using this manual
• Typographical conventions
• Examples of syntax descriptions
• OpenEdge messages
• Third party acknowledgements
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

Part I, Database Basics

Chapter 1, “Creating and Deleting Databases”

Describes how to create and delete OpenEdge databases.

Chapter 2, “OpenEdge RDBMS Limits”

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

Chapter 3, “Starting Up and Shutting Down”

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

Part II, Protecting Your Data

Chapter 4, “Backup Strategies”

Discusses various approaches to backing up your database.

Chapter 5, “Backing Up a Database”

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

Chapter 6, “Recovering a Database”

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

Chapter 7, “After-imaging”

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

Chapter 8, “Maintaining Security”

Describes how to implement database security, including assigning user IDs and designating database administrators.
Chapter 9, “Auditing”

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

Chapter 10, “Transparent Data Encryption”

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.

Chapter 11, “Replicating Data”

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

Chapter 12, “Failover Clusters”

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

Chapter 13, “Distributed Transaction Processing”

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

Part III, Maintaining and Monitoring Your Database

Chapter 14, “Managing Performance”

Discusses how to monitor tune database performance.

Chapter 15, “Maintaining Database Structure”

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

Chapter 16, “Dumping and Loading”

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

Chapter 17, “Logged Data”

Examines the process of logging significant database events.

Part IV, Reference

Chapter 18, “Startup and Shutdown Commands”

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

Chapter 19, “Database Startup Parameters”

Lists and details the OpenEdge RDBMS startup parameters.

Chapter 20, “PROMON Utility”

Details the PROMON Utility used for monitoring your database.
Chapter 21, “PROUTIL Utility”
Details the PROUTIL Utility used for maintaining your database.

Chapter 22, “PROSTRCT Utility”
Details the PROSTRCT Utility used for creating and updating the physical structure of your database.

Chapter 23, “RFUTIL Utility”
Details the RFUTIL Utility used for managing after imaging.

Chapter 24, “Other Database Administration Utilities”
Details other database utilities including PROBKUP, PROREST, PROCOPY, PRODEL, and PROLOG.

Chapter 25, “SQL Utilities”
Details the utilities used for maintaining your database for use with SQL.

Chapter 26, “Virtual System Tables”
Describes the Virtual System Tables that allow ABL and SQL applications to examine the status of a database and monitor its performance.

Using this manual
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.

- 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.

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

- 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.

For the latest documentation updates see the OpenEdge Product Documentation Overview page on PSDN: http://communities.progress.com/pcom/docs/DOC-16074.

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.
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 manual uses the following typographical conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Bold typeface indicates commands or characters the user types, provides emphasis, or the names of user interface elements.</td>
</tr>
<tr>
<td>Italic</td>
<td>Italic typeface indicates the title of a document, or signifies new terms.</td>
</tr>
<tr>
<td>Convention</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>SMALL, BOLD CAPITAL LETTERS</td>
<td>Small, bold capital letters indicate OpenEdge key functions and generic keyboard keys; for example, GET and CTRL.</td>
</tr>
<tr>
<td>KEY1+KEY2</td>
<td>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.</td>
</tr>
<tr>
<td>KEY1 KEY2</td>
<td>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.</td>
</tr>
<tr>
<td><strong>Syntax:</strong></td>
<td></td>
</tr>
<tr>
<td>Fixed width</td>
<td>A fixed-width font is used in syntax statements, code examples, system output, and filenames.</td>
</tr>
<tr>
<td>Fixed-width italics</td>
<td>Fixed-width italics indicate variables in syntax statements.</td>
</tr>
<tr>
<td>Fixed-width bold</td>
<td>Fixed-width bold indicates variables with special emphasis.</td>
</tr>
<tr>
<td>UPPERCASE fixed width</td>
<td>Uppercase words are ABL keywords. Although these are always shown in uppercase, you can type them in either uppercase or lowercase in a procedure.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Large brackets indicate the items within them are optional.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Small brackets are part of ABL.</td>
</tr>
<tr>
<td>{ }</td>
<td>Large braces indicate the items within them are required. They are used to simplify complex syntax diagrams.</td>
</tr>
<tr>
<td>{ }</td>
<td>Small braces are part of ABL. For example, a called external procedure must use braces when referencing arguments passed by a calling procedure.</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td>Ellipses indicate repetition: you can choose one or more of the preceding items.</td>
</tr>
</tbody>
</table>
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:

   \[
   \text{OpenEdge-install-dir/bin/pro}
   \]

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.

Third party acknowledgements

OpenEdge includes AdventNet - Agent Toolkit licensed from AdventNet, Inc.
http://www.adventnet.com. All rights to such copyright material rest with AdventNet.

OpenEdge includes ANTLR (Another Tool for Language Recognition) software Copyright ©
2003-2006, Terence Parr All rights reserved. Neither the name of the author nor the names of
its contributors may be used to endorse or promote products derived from this software without
specific prior written permission. Software distributed on an “AS IS” basis, WITHOUT
WARRANTY OF ANY KIND, either express or implied. See the License for the specific
language governing rights and limitations under the License agreement that accompanies the
product.

OpenEdge includes software developed by the Apache Software Foundation
(http://www.apache.org/). Copyright © 1999 The Apache Software Foundation. All rights
reserved (Xerces C++ Parser (XML) and Xerces2 Java Parser (XML)); Copyright © 1999-2002
The Apache Software Foundation. All rights reserved (Xerces Parser (XML)); and Copyright ©
2000-2003 The Apache Software Foundation. All rights reserved (Ant). The names “Apache,”
“Xerces,” “ANT,” and “Apache Software Foundation” must not be used to endorse or promote
products derived from this software without prior written permission. Products derived from
this software may not be called “Apache”, nor may “Apache” appear in their name, without
prior written permission of the Apache Software Foundation. For written permission, please
contact apache@apache.org. Software distributed on an “AS IS” basis, WITHOUT
WARRANTY OF ANY KIND, either express or implied. See the License for the specific
language governing rights and limitations under the License agreement that accompanies the
product.

OpenEdge includes Concurrent Java software Copyright 1994-2000 Sun Microsystems, Inc. All
Rights Reserved. -Neither the name of or trademarks of Sun may be used to endorse or promote
products including or derived from the Java Software technology without specific prior written
permission; and Redistributions of source or binary code must contain the above copyright
notice, this notice and the following disclaimers: This software is provided "AS IS," without a
warranty of any kind. ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS
AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF
MERCHANTABLE, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE HEREBY EXCLUDED. SUN MICROSYSTEMS, INC. AND ITS LICENSORS SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THE SOFTWARE OR ITS DERIVATIVES. IN NO EVENT WILL SUN MICROSYSTEMS, INC. OR ITS LICENSORS BE LIABLE FOR ANY LOST REVENUE, PROFIT OR DATA, OR FOR DIRECT, INDIRECT, SPECIAL, CONSEQUENTIAL, INCIDENTAL OR PUNITIVE DAMAGES, HOWEVER CAUSED AND REGARDLESS OF THE THEORY OF LIABILITY, ARISING OUT OF THE USE OF OR INABILITY TO USE SOFTWARE, EVEN IF SUN MICROSYSTEMS, INC. HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

OpenEdge includes DataDirect software Copyright © 1991-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for JDBC Type 4 driver); Copyright © 1993-2009 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for JDBC); Copyright © 1988-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect for ODBC); and Copyright © 1988-2007 Progress Software Corporation and/or its subsidiaries or affiliates. All Rights Reserved. (DataDirect Connect64 for ODBC).

OpenEdge includes DataDirect Connect for ODBC and DataDirect Connect64 for ODBC software, which include ICU software 1.8 and later - Copyright © 1995-2003 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.

OpenEdge includes DataDirect Connect for ODBC and DataDirect Connect64 for ODBC software, which include software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/). Copyright © 1998-2006 The OpenSSL Project. All rights reserved. And Copyright © 1995-1998 Eric Young (eay@cryptsoft.com). All rights reserved.

OpenEdge includes DataDirect products for the Microsoft SQL Server database which contain a licensed implementation of the Microsoft TDS Protocol.

OpenEdge includes software authored by David M. Gay. Copyright © 1991, 2000, 2001 by Lucent Technologies (dtoa.c); Copyright © 1991, 1996 by Lucent Technologies (g_fmt.c); and Copyright © 1991 by Lucent Technologies (rnd_prod.s). Permission to use, copy, modify, and distribute this software for any purpose without fee is hereby granted, provided that this entire notice is included in all copies of any software which is or includes a copy or modification of this software and in all copies of the supporting documentation for such software. THIS SOFTWARE IS BEING PROVIDED "AS IS", WITHOUT ANY EXPRESS OR IMPLIED WARRANTY. IN PARTICULAR, NEITHER THE AUTHOR NOR LUCENT MAKES ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR PURPOSE.

OpenEdge includes software authored by David M. Gay. Copyright © 1998-2001 by Lucent Technologies All Rights Reserved (decstrtod.c; strtodg.c); Copyright © 1998, 2000 by Lucent Technologies All Rights Reserved (decstrtof.c; strtord.c); Copyright © 1998 by Lucent
Preface

Technologies All Rights Reserved (dmisc.c; gdtoa.h; gethex.c; gmisc.c; sum.c); Copyright © 1998, 1999 by Lucent Technologies All Rights Reserved (gdtoa.c; misc.c; smisc.c; ulp.c); Copyright © 1998-2000 by Lucent Technologies All Rights Reserved (gdtoaimp.h); Copyright © 2000 by Lucent Technologies All Rights Reserved (hd_init.c). Full copies of these licenses can be found in the installation directory, in the c:/OpenEdge/licenses folder. Permission to use, copy, modify, and distribute this software and its documentation for any purpose and without fee is hereby granted, provided that the above copyright notice appear in all copies and that both that the copyright notice and this permission notice and warranty disclaimer appear in supporting documentation, and that the name of Lucent or any of its entities not be used in advertising or publicity pertaining to distribution of the software without specific, written prior permission. LUCENT DISCLAIMS ALL WARRANTIES WITH REGARD TO THIS SOFTWARE, INCLUDING ALL IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL LUCENT OR ANY OF ITS ENTITIES BE LIABLE FOR ANY SPECIAL, INDIRECT OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.

OpenEdge includes http package software developed by the World Wide Web Consortium. Copyright © 1994-2002 World Wide Web Consortium, (Massachusetts Institute of Technology, European Research Consortium for Informatics and Mathematics, Keio University). All rights reserved. This work is distributed under the W3C® Software License [http://www.w3.org/Consortium/Legal/2002/copyright-software-20021231] in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

OpenEdge includes ICU software 1.8 and later - Copyright © 1995-2003 International Business Machines Corporation and others All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation.


OpenEdge includes Infragistics NetAdvantage for .NET v2009 Vol 2 Copyright © 1996-2009 Infragistics, Inc. All rights reserved.

OpenEdge includes JSTL software Copyright 1994-2006 Sun Microsystems, Inc. All Rights Reserved. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes OpenSSL software developed by the OpenSSL Project for use in the OpenSSL Toolkit (http://www.openssl.org/). Copyright © 1998-2007 The OpenSSL Project. All rights reserved. This product includes cryptographic software written by Eric Young (eay@cryptsoft.com). This product includes software written by Tim Hudson (tjh@cryptsoft.com). Copyright © 1995-1998 Eric Young (eay@cryptsoft.com) All rights reserved. The names "OpenSSL Toolkit" and "OpenSSL Project" must not be used to endorse or promote products derived from this software without prior written permission. For written
permission, please contact openssl-core@openssl.org. Products derived from this software may not be called "OpenSSL" nor may "OpenSSL" appear in their names without prior written permission of the OpenSSL Project. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product.

OpenEdge includes Quartz Enterprise Job Scheduler software Copyright © 2001-2003 James House. All rights reserved. Software distributed on an “AS IS” basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the product. This product uses and includes within its distribution, software developed by the Apache Software Foundation (http://www.apache.org/).

OpenEdge includes code licensed from RSA Security, Inc. Some portions licensed from IBM are available at http://oss.software.ibm.com/icu4j/.

OpenEdge includes the RSA Data Security, Inc. MD5 Message-Digest Algorithm. Copyright ©1991-2, RSA Data Security, Inc. Created 1991. All rights reserved.

OpenEdge includes Sonic software, which includes software developed by Apache Software Foundation (http://www.apache.org/). Copyright © 1999-2000 The Apache Software Foundation. All rights reserved. The names “Ant”, “Axis”, “Xalan,” “FOP,” “The Jakarta Project”, “Tomcat”, “Xerces” and/or “Apache Software Foundation” must not be used to endorse or promote products derived from the Product without prior written permission. Any product derived from the Product may not be called “Apache”, nor may “Apache” appear in their name, without prior written permission. For written permission, please contact apache@apache.org.

OpenEdge includes Sonic software, which includes software Copyright © 1999 CERN - European Organization for Nuclear Research. Permission to use, copy, modify, distribute and sell this software and its documentation for any purpose is hereby granted without fee, provided that the above copyright notice appear in all copies and that both that copyright notice and this permission notice appear in supporting documentation. CERN makes no representations about the suitability of this software for any purpose. It is provided "as is" without expressed or implied warranty.

OpenEdge includes Sonic software, which includes software developed by ExoLab Project (http://www.exolab.org/). Copyright © 2000 Intalio Inc. All rights reserved. The names “Castor” and/or “ExoLab” must not be used to endorse or promote products derived from the Products without prior written permission. For written permission, please contact info@exolab.org. Exolab, Castor and Intalio are trademarks of Intalio Inc.

OpenEdge includes Sonic software, which includes software developed by IBM. Copyright © 1995-2003 International Business Machines Corporation and others. All rights reserved. Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, provided that the above copyright notice(s) and this permission notice appear in all copies of the Software and that both the above copyright notice(s) and this permission notice appear in supporting documentation. Software distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License agreement that accompanies the
product. Except as contained in this notice, the name of a copyright holder shall not be used in
advertising or otherwise to promote the sale, use or other dealings in this Software without prior
written authorization of the copyright holder.

OpenEdge includes Sonic software, which includes the JMX Technology from Sun
Microsystems, Inc. Use and Distribution is subject to the Sun Community Source License

OpenEdge includes Sonic software, which includes software developed by the ModelObjects
rights reserved. The name “ModelObjects” must not be used to endorse or promote products
derived from this software without prior written permission. Products derived from this
software may not be called “ModelObjects”, nor may “ModelObjects” appear in their name,
without prior written permission. For written permission, please contact
djacobs@modelobjects.com.

OpenEdge includes Sonic software, which includes code licensed from Mort Bay Consulting
Pty. Ltd. The Jetty Package is Copyright © 1998 Mort Bay Consulting Pty. Ltd. (Australia) and
others.

OpenEdge includes Sonic software, which includes files that are subject to the Netscape Public
License Version 1.1 (the “License”); you may not use this file except in compliance with the
License. You may obtain a copy of the License at http://www.mozilla.org/NPL/. Software
distributed under the License is distributed on an “AS IS” basis, WITHOUT WARRANTY OF
ANY KIND, either express or implied. See the License for the specific language governing
rights and limitations under the License. The Original Code is Mozilla Communicator client
code, released March 31, 1998. The Initial Developer of the Original Code is Netscape
Communications Corporation. Portions created by Netscape are Copyright 1998-1999
Netscape Communications Corporation. All Rights Reserved.

OpenEdge includes Sonic software, which includes software developed by the University
Corporation for Advanced Internet Development http://www.ucaid.edu Internet2 Project.
Copyright © 2002 University Corporation for Advanced Internet Development, Inc. All rights
reserved. Neither the name of OpenSAML nor the names of its contributors, nor Internet2, nor
the University Corporation for Advanced Internet Development, Inc., nor UCAID may be used
to endorse or promote products derived from this software and products derived from this
software may not be called OpenSAML, Internet2, UCAID, or the University Corporation for
Advanced Internet Development, nor may OpenSAML appear in their name without prior
written permission of the University Corporation for Advanced Internet Development. For
written permission, please contact opensaml@opensaml.org.

OpenEdge includes the UnixWare platform of Perl Runtime authored by Kiem-Phong Vo and
David Korn. Copyright © 1991, 1996 by AT&T Labs. Permission to use, copy, modify, and
distribute this software for any purpose without fee is hereby granted, provided that this entire
notice is included in all copies of any software which is or includes a copy or modification of
this software and in all copies of the supporting documentation for such software. THIS
SOFTWARE IS BEING PROVIDED “AS IS”, WITHOUT ANY EXPRESS OR IMPLIED
WARRANTY. IN PARTICULAR, NEITHER THE AUTHORS NOR AT&T LABS MAKE
ANY REPRESENTATION OR WARRANTY OF ANY KIND CONCERNING THE
MERCHANTABILITY OF THIS SOFTWARE OR ITS FITNESS FOR ANY PARTICULAR
PURPOSE.

OpenEdge includes Vermont Views Terminal Handling Package software developed by
OpenEdge includes XML Tools, which includes versions 8.9 of the Saxon XSLT and XQuery Processor from Saxonica Limited (http://www.saxonica.com/) which are available from SourceForge (http://sourceforge.net/projects/saxon/). The Original Code of Saxon comprises all those components which are not explicitly attributed to other parties. The Initial Developer of the Original Code is Michael Kay. Until February 2001 Michael Kay was an employee of International Computers Limited (now part of Fujitsu Limited), and original code developed during that time was released under this license by permission from International Computers Limited. From February 2001 until February 2004 Michael Kay was an employee of Software AG, and code developed during that time was released under this license by permission from Software AG, acting as a "Contributor". Subsequent code has been developed by Saxonica Limited, of which Michael Kay is a Director, again acting as a "Contributor". A small number of modules, or enhancements to modules, have been developed by other individuals (either written especially for Saxon, or incorporated into Saxon having initially been released as part of another open source product). Such contributions are acknowledged individually in comments attached to the relevant code modules. All Rights Reserved. The contents of the Saxon files are subject to the Mozilla Public License Version 1.0 (the "License"); you may not use these files except in compliance with the License. You may obtain a copy of the License at http://www.mozilla.org/MPL/ and a copy of the license can also be found in the installation directory, in the c:/OpenEdge/licenses folder. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License.

OpenEdge includes Xs3P v1.1.3. The contents of this file are subject to the DSTC Public License (DPL) Version 1.1 (the "License"); you may not use this file except in compliance with the License. A copy of the license can be found in the installation directory, in the c:/OpenEdge/licenses folder. Software distributed under the License is distributed on an "AS IS" basis, WITHOUT WARRANTY OF ANY KIND, either express or implied. See the License for the specific language governing rights and limitations under the License. The Original Code is xs3p. The Initial Developer of the Original Code is DSTC. Portions created by DSTC are Copyright © 2001, 2002 DSTC Pty Ltd. All rights reserved.

OpenEdge includes YAJL software Copyright 2007, Lloyd Hilaiel. Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met: 1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer. 2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution. 3. Neither the name of Lloyd Hilaiel nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission. THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
Database Basics

Chapter 1, Creating and Deleting Databases

Chapter 2, OpenEdge RDBMS Limits

Chapter 3, Starting Up and Shutting Down
Creating and Deleting Databases

This chapter describes the methods to create and delete an OpenEdge® database, as detailed in the following sections:

- Ways to create an OpenEdge database
- Creating a database with PROSTRCT CREATE
- Creating a database with the PRODB utility
- Creating a database with a Data tool
- Using an OpenEdge database as an Actional repository
- Converting a Progress Version 9 database to OpenEdge Release 10
- Migrating from Release 10.1A
- 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 on the command line
- The Data Dictionary tool if you are using a graphical interface or a character interface
- The Data Administration tool if you are using a graphical interface
- The PROCOPY utility
- The PROREST utility
- A Progress® Version 9 database by converting it to an OpenEdge Release 10 database

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.
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

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:

```plaintext
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 (*).
Table 1–1 explains the value of each of the 8 tokens in a structure description file.

### Table 1–1: ST file tokens

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>type</td>
<td>Indicates the type of storage area. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• a — After-image area</td>
</tr>
<tr>
<td></td>
<td>• b — Before-image area</td>
</tr>
<tr>
<td></td>
<td>• d — Schema and application data areas</td>
</tr>
<tr>
<td></td>
<td>• t — Transaction log area</td>
</tr>
<tr>
<td>areaname</td>
<td>Name of the storage area.</td>
</tr>
<tr>
<td>areanum</td>
<td>Number of the storage area.</td>
</tr>
<tr>
<td></td>
<td><strong>Note:</strong> 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.</td>
</tr>
<tr>
<td>recsPerBlock</td>
<td>Number of database records in each database block. Valid values are 1, 2, 4, 8, 26, 32, 64, 128, and 256.</td>
</tr>
<tr>
<td></td>
<td><strong>Notes:</strong></td>
</tr>
<tr>
<td></td>
<td>• This token only applies to areas 7 through 32,000.</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td>blcksPerCluster</td>
<td>Number of database blocks in each cluster. Possible values are: 1, 8, 64, or 512.</td>
</tr>
<tr>
<td></td>
<td><strong>Notes:</strong></td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• This token only applies to areas 7 through 32,000.</td>
</tr>
<tr>
<td>path</td>
<td>Absolute or relative pathname of each extent.</td>
</tr>
<tr>
<td>extentType</td>
<td>Indicates whether the extent is fixed (f) or variable (v). If the extent type token is not specified, the extent is variable.</td>
</tr>
<tr>
<td>size</td>
<td>Size of an extent in kilobytes. This value must be a multiple of 16 times your database block size.</td>
</tr>
</tbody>
</table>
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 \((16 \times database\text{-}blocksize)\). 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 \((16 \times database\text{-}blocksize)\). 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.
Table 1–2 shows how the extent size changes based on the database block size.

### Table 1–2: Calculating extent size

<table>
<thead>
<tr>
<th>Database block size</th>
<th>Formula</th>
<th>Extent size</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2*(16 * 1)</td>
<td>32K</td>
</tr>
<tr>
<td></td>
<td>3*(16 * 1)</td>
<td>48K</td>
</tr>
<tr>
<td>2</td>
<td>1*(16 * 2)</td>
<td>32K</td>
</tr>
<tr>
<td></td>
<td>2*(16 * 2)</td>
<td>64K</td>
</tr>
<tr>
<td></td>
<td>3*(16 * 2)</td>
<td>96K</td>
</tr>
<tr>
<td>4</td>
<td>1*(16 * 4)</td>
<td>64K</td>
</tr>
<tr>
<td></td>
<td>2*(16 * 4)</td>
<td>128K</td>
</tr>
<tr>
<td></td>
<td>3*(16 * 4)</td>
<td>192K</td>
</tr>
<tr>
<td>8</td>
<td>1*(16 * 8)</td>
<td>128K</td>
</tr>
<tr>
<td></td>
<td>2*(16 * 8)</td>
<td>256K</td>
</tr>
<tr>
<td></td>
<td>3*(16 * 8)</td>
<td>384K</td>
</tr>
</tbody>
</table>

- **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.

**Notes:** 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. 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
```

For more information on enabling large file processing, see the “PROUTIL ENABLELARGEFILES qualifier” section on page 21–54.

Create a database structure extent

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

**Syntax**

```
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:

   ```bash
   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 as follows:

   ```bash
   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, .ti, and .dn 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. Use the PROCOPY utility to copy the system tables from an empty OpenEdge database into the void database you created with PROSTRCT CREATE as shown:

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

   Where emptyn is the source database and db-name is the target database. n indicates the block size of the OpenEdge-supplied empty.db. See the “PRODB utility” section on page 24–23 for more information about the empty and emptyn databases.

2. Use the PROSTRCT LIST utility to verify that you have the correct database files in the correct locations, as shown:

   ```
   prostrct list db-name
   ```

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

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

   **Note:** 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 was 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 the “PRODB utility” section on page 24–23 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.

**Note:** See Chapter 15, “Maintaining Database Structure,” for information about changing pathname conventions when adding storage areas and extents to a structure description file.
**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/Block 32, 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
```

See *OpenEdge Getting Started: Database Essentials* for an explanation of relative- and absolute-pathnames.
Creating a database with a Data tool

The Data Administration tool provides a graphical interface for administering OpenEdge databases. The Data Dictionary tool allows you to administer OpenEdge databases in a character interface. See the Data Administration tool’s online help for details on using it to create databases. For details about using the Data Dictionary in a character interface to create databases, see OpenEdge Development: Basic Database Tools.
Using an OpenEdge database as an Actional repository

Progress Actional Enterprise is a server product used to monitor operations in a widely distributed Service Oriented Architecture (SOA) environment. The primary component of the product is a server, called the Actional Management Server (AMS), that collects, analyzes, and displays operational data.

Starting with Version 8.0.3, the AMS includes the ability to use the OpenEdge database as the report database that maintains statistical data for the Actional Management Server and the Actional Agent. Creating and tuning a database is unique for each environment; this section presents some starting points for you to consider. You should refine your original settings as you monitor the operation of your environment.

Before creating an OpenEdge database for this purpose, you should review the section on setting up external databases in the Actional Management Server Guide. This includes specific instructions for configuring Actional to use an OpenEdge database. For more information on OpenEdge support for Actional, see OpenEdge Getting Started: Application and Integration Services.

Note: As of OpenEdge Release 10.2B, you only need to register OpenEdge_install_dir\java\openedge.jar for the Classpath in Actional. If you use an OpenEdge database from a previous release, you also need to register the base.jar and util.jar.

Your Actional Management Server installation includes a SQL script to add the schema for the Actional repository to your database. You can find the scripts in the <Actional-install>\config\SQL\OpenEdge directory.

When creating and tuning an OpenEdge database to serve as an Actional repository, consider the following points:

- The database must be SQL-enabled. Actional uses SQL to access its repository.
- The database administrator account must have DBA and RESOURCE access.
- Choose a database blocksize of 4096, at a minimum.
- Actional requires a longer PROSQL_LOCKWAIT_TIMEOUT than the default. Set this SQL environment variable to 30 to start. You can tune it as appropriate later.
- An enterprise SOA environment can create a large volume of I/O to an Actional repository. Choose the startup parameters for your environment accordingly.
Converting a Progress Version 9 database to OpenEdge Release 10

The PROUTIL CONV910 utility converts the schema of a Progress Version 9 database to an OpenEdge Release 10 database.

To convert the schema of a Progress Version 9 database to an OpenEdge Release 10 database:


   **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 revert to the backup copy of your database and begin the conversion again.

2. Disable after-imaging and two-phase commit. You should disable after-imaging and two-phase commit before starting the conversion; however, if you forget to do so, PROUTIL will disable after-imaging and two-phase commit for you. PROUTIL issues an informational message when it disables after-imaging and/or two-phase commit.

3. Truncate your before-image file. PROUTIL will not convert your Version 9 database schema if you do not truncate the before-image file before you start the conversion.

4. Disable replication, if applicable. PROUTIL cannot convert a Version 9 database if replication is enabled.

5. Verify that your Version 9 backup exists, then install OpenEdge Release 10 following the instructions in *OpenEdge Getting Started: Installation and Configuration*.

6. Run PROUTIL CONV910 to convert your database, as shown:

   ```
   proutil db-name -C conv910
   ```

7. After you have successfully converted your database, back up your OpenEdge Release 10 database.

   You should back up your Release 10 database in case it is damaged due to some failure. If you have only a Progress Version 9 backup of your database, then you would need to go through the conversion process again to convert your Version 9 backup to an OpenEdge Release 10 database.
Creating and Deleting Databases

Using the Schema Mover after conversion

After converting a Progress Version 9 database to an OpenEdge Release 10 database, it is possible to separate the schema and user data by using PROUTIL dump and load or bulkload qualifiers, the Database Administration Tool, 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 the area can be 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. Enter the following command to begin the schema move, where `dbname` is the name of the converted database:

   ```
   proutil dbname -C mvsch
   ```
As shown in Figure 1–1, after you convert your database to OpenEdge Release 10, Area 6 (NewDB.db) contains both the database’s schema and data. When you initiate the schema move by entering the syntax shown in Step 3, PROUTIL finds the next unused data area (starting at Area 7) and creates the target data area and the target data area’s extent. After creating the target data area, PROUTIL moves the schema from NewDB.db to NewDB_7.d1. All the schema records and indexes are deleted from NewDB.db. PROUTIL opens the .d1 files of the NewDB.db and NewDB_7.d1 and then updates and switches the master and area blocks. After the switch, the areas and their extents are renamed. Now that the schema is gone from the “Old Default Area,” you can truncate it and recover any unused space.

**Figure 1–1: How the Schema Mover works**

<table>
<thead>
<tr>
<th>Without Schema Mover</th>
<th>With Schema Mover</th>
</tr>
</thead>
<tbody>
<tr>
<td>OldDB.db</td>
<td>OldDB.db</td>
</tr>
<tr>
<td>OldDB.d1</td>
<td>OldDB.d1</td>
</tr>
<tr>
<td></td>
<td>PROUTIL CONV910</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NewDB.db</td>
<td>NewDB.db</td>
</tr>
<tr>
<td>NewDB.d1</td>
<td>NewDB.d1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Dump and Load; Index rebuild</td>
<td>Schema moved to next available area</td>
</tr>
<tr>
<td>NewDB_7.d1</td>
<td>NewDB_7.d1</td>
</tr>
<tr>
<td>20 GB</td>
<td>60 GB</td>
</tr>
<tr>
<td>OldDB.d1</td>
<td>OldDB.d1</td>
</tr>
<tr>
<td></td>
<td>PROUTIL CONV910</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>NewDB.db</td>
<td>NewDB.db</td>
</tr>
<tr>
<td>NewDB.d1</td>
<td>NewDB.d1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Schema moved to next available area</td>
<td>Areas renamed</td>
</tr>
<tr>
<td>NewDB_7.d1</td>
<td>NewDB_7.d1</td>
</tr>
<tr>
<td>“Old default area”</td>
<td>“Old default area”</td>
</tr>
<tr>
<td>NewDB_7.d1</td>
<td>NewDB_7.d1</td>
</tr>
<tr>
<td>60 GB</td>
<td>60 GB</td>
</tr>
<tr>
<td>Unused space in “Old default area” can now be recovered</td>
<td>Unused space in “Old default area” can now be recovered</td>
</tr>
</tbody>
</table>

**Note:** Data in the “Old Default Area” that is not moved prior to area truncation will be lost.
Migrating from Release 10.1A

Release 10.1B introduced many internal changes that require steps to migrate databases to versions at or above this release, and prevents the interoperability with prior releases, as discussed in the following sections:

- Database migration and interoperability
- RDBMS client connections from prior releases
- Reverting your database to Release 10.1A

Database migration and interoperability

Release 10.1B and higher cannot open a 10.1A database that has:

- An untruncated BI file
- After-imaging enabled
- JTA enabled
- Two-phase commit enabled

Therefore, prior to opening an existing 10.1A database with a higher release, you must:

- Truncate the BI file with the 10.1A PROUTIL
- Disable after-imaging with the 10.1A RFUTIL
- Disable JTA with the 10.1A PROUTIL
- Disable Two-phase commit with the 10.1A PROUTIL

Once you have migrated a database to Release 10.1B or higher, you cannot open it with 10.1A without first reverting its format. Some databases can be reverted with PROUTIL REVERT, but not all databases can be reverted without a dump and load. For information on PROUTIL REVERT, see the “Reverting your database to Release 10.1A” section on page 1–22.
Other formatting changes that result in incompatibility include:

- **Backup formats** — Backup formats are incompatible between Release 10.1A and Release 10.1B and higher. The following restrictions exist:
  - You **cannot** back up a 10.1A database with the 10.1B or higher PROBKUP.
    
    If you attempt to back up a 10.1A database with the 10.1B PROBKUP, the utility will use the 10.1A PROBKUP if it is available. The following sample output shows the results of backing up a 10.1A database with 10.1B PROBKUP:

    ```
    $ probkup my_101a_db my_101a_db.bck
    Database sp_101a uses 32-bit DBKEYs.
    The 10.1A dbutil will be used to do the backup.
    ... 
    ```

    The 10.1A utilities are found in the directory `OpenEdge-install-dir/bin/101dbutil`. 

    - You **cannot** back up a 10.1B or higher database with the 10.1A PROBKUP.
    - You **cannot** restore a 10.1B or higher backup with the 10.1A PROREST.
    - You **can** restore a 10.1A backup with the 10.1B or higher PROREST provided that:
      - The database contains no before-image blocks
      - The database contains no transaction log blocks

    The target database will be a database of the current release. If the target exists, and is a 10.1A database, it will be converted to 10.1C the current format as part of the PROREST restore.

- **After-image formats** — After-image formats are incompatible. You cannot apply after-image areas across releases.

- **Before-image formats** — Before-image formats are incompatible. PROUTIL TRUNCATE BI is incompatible between Release 10.1A and Release 10.1B and higher. You must truncate the BI file of your 10.1A database with the 10.1A PROUTIL before you can open the database with Release 10.1B or higher.
RDBMS client connections from prior releases

Clients from Releases 10.1A and 10.0B can connect to a Release 10.1B and higher database. However, their access is limited as follows:

- An older client cannot access a row that is addressed beyond the 2-billion row limit.
- An older client cannot update or delete any row or field that is referenced by a large index key.
- An older client cannot reference a sequence with a value greater than can be stored in a 32-bit integer.

When a client tries to access data beyond the 2-billion row limit or a large sequence value, the server disconnects the client from the database with this error:

Disconnected by the server, code 44.

When a client tries to update a large index key, the client is shut down with this error:

SYSTEM ERROR: fillkey: error formatting existing key.

During the shut down, the client generates a core dump and stack trace. When the 2-billion row limit will be exceeded is not predictable, and is not limited to the actual data. For example, you could encounter this error by expanding an area while creating a record or during the retrieval of an existing record with a ROWID beyond the 2-billion row limit.

Reverting your database to Release 10.1A

Due to the limited interoperability between databases from Release 10.1A and Release 10.1B and forward, the PRUTIL REVERT utility is available to revert to the format of a 10.1A database, for databases that meet certain criteria.

PRUTIL REVERT cannot revert a database if any of the following conditions exist:

- The database was created with Release 10.1B or higher.
- The schema contains any INT64 fields.
- The database has enabled support for large key entries for indexes.
- The database has enabled support for 64-bit sequences.
- The database has a Type II area with a high water mark utilizing 64-bit DB Keys; this includes a LOB with segments utilizing 64-bit block values.
- The database has area numbers greater than 1000.

The syntax of PRUTIL REVERT is:

Syntax

proutil db-name -C revert
PROUTIL REVERT analyzes the database and determines if it can be reverted with this utility. If PROUTIL REVERT cannot revert the database, you will need to dump and load the database after removing or restructuring any aspects of your database that cannot be reverted. See the “Reverting your database manually” section on page 1–24 for more information.

**Caution:** PROUTIL REVERT is a non-recoverable utility. If PROUTIL REVERT terminates abnormally, you must restore your database from backup.

PROUTIL REVERT runs against an offline database as follows:

1. It determines that the user has sufficient privilege to execute the command. The privilege check is limited to file system access to the database.

2. It analyzes the features of the database to determine if the database can be reverted by the utility. If not, the utility issues messages indicating why the database cannot be reverted, and exits.

   The following sample output is from an attempt to revert a database that does not meet the reversion requirements:

<table>
<thead>
<tr>
<th>Revert Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
</tr>
<tr>
<td>----------------</td>
</tr>
<tr>
<td>64 Bit DBKEYS</td>
</tr>
<tr>
<td>Large Keys</td>
</tr>
<tr>
<td>64 Bit Sequences</td>
</tr>
</tbody>
</table>

   Revert: The database actively supports Large Key Indexes. (13749)
   Revert: The database can not be reverted. (13750)
   Revert: The database contains 64-bit Sequences. (13751)
   Revert: The database can not be reverted. (13750)

3. It prompts the user to confirm that the database has been backed up.

4. It performs the physical fixes necessary to revert the database. Fixes include the following:
   - Reverting values in the database master block to 10.1A format
   - Reverting object block values to 10.1A format
   - Removing current VSTs and adding the 10.1A VSTs
   - Removing all current settings from the feature mask

5. It disables 2-Phase commit, JTA, Replication, After-imaging, and AI Management, if active.

6. It truncates the BI file.
PROUTIL REVERT provides informational messages updating the progress of the revert process. The following sample output is from a successful revert:

```
Revert Utility

Feature                           Enabled  Active
--------------------------------  -------  ------
Database Auditing                 Yes      Yes
64 Bit DBKEYS                     Yes      Yes

Revert: Have you backed up your database..... Type y to continue (y/n).
(13754)  
y
Revert: Database beginning reversion process. (13755)
VST Table Deletion has begun. Please standby. (6876)
VST Table Deletion has completed successfully. (6885)
Revert: 64 Bit Dbkey has been disabled for this database.
After-image disabled. (846)
Revert: Truncating the Bi File.
VST Table Deletion has begun. Please standby. (6876)
VST Table Deletion has completed successfully. (6885)
Adding VST file: _Connect. (6875)
Adding VST file: _MstrBlk. (6875)
Adding VST file: _DbStatus. (6875)
.
.
Adding VST file: _AreaThreshold. (6875)
Total number of Virtual System Tables is 45. (6250)
Revert: Database ending reversion process.
```

When PROUTIL REVERT completes successfully, the database is in 10.1A format. You should perform a backup with 10.1A; previous backups in the current release format are incompatible with the reverted database.

For more information on PROUTIL REVERT, see the “PROUTIL REVERT qualifier” section on page 21–90.

**Reverting your database manually**

If PROUTIL REVERT cannot revert your database, you must dump and reload it, after ensuring that the contents are backward-compatible. This will require that you:

- Remove any INT64 fields from the schema
- Delete any indexes with keys wider than the supported 10.1A maximum of 200 characters
- Remove any sequences that have values larger than a 32-bit signed integer
- Break up your tables so that no table has more than 2 billion rows
- Remove or renumber areas so that there are no areas with an area number greater than 1000
Database conversion utilities

OpenEdge Release 10 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

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 the “PRODB utility” section on page 24–23 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.
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/Block 32, 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.
OpenEdge RDBMS Limits

This chapter lists the limits you need to know when configuring a database and supporting an application development environment, as described in the following sections:

- 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. Table 2–1 describes the area types.

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

Table 2–1: Storage area types

<table>
<thead>
<tr>
<th>Reserved area name</th>
<th>Contents</th>
<th>File extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Control</td>
<td>Physical Database Structure</td>
<td>.db</td>
</tr>
<tr>
<td>Primary Recovery</td>
<td>Recovery Log Data</td>
<td>.bn</td>
</tr>
<tr>
<td>Transaction Log</td>
<td>Two-phase Commit Transaction Log</td>
<td>.tn</td>
</tr>
<tr>
<td>After Image</td>
<td>After-image Log Data</td>
<td>.an</td>
</tr>
<tr>
<td>Schema</td>
<td>Schema Data</td>
<td>.dn</td>
</tr>
<tr>
<td>–</td>
<td>Application Data</td>
<td>.dn</td>
</tr>
</tbody>
</table>

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:

\[
\text{Maximum area size} = \text{maximum-number-of-extents-per-area} \times \text{maximum-extent-size} \\
= 1024 \times 1\text{TB} \\
= 2^{10} \times 2^{40} \\
= 2^{50} \text{ bytes} \\
= 1\text{PB}
\]

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

\[
\text{Maximum records per area} = \text{Maximum area size} \times \frac{\text{records-per-block}}{\text{block size}}
\]
Table 2–2 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.

Table 2–2: Maximum rows per Type II storage area (approximate)  

<table>
<thead>
<tr>
<th>Database block size</th>
<th>Records per block</th>
<th>Records per area</th>
<th>Approximate maximum records per area (in M)¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>8192 bytes (8K)</td>
<td>1</td>
<td>2&lt;sup&gt;37&lt;/sup&gt;</td>
<td>137,439 M</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2&lt;sup&gt;38&lt;/sup&gt;</td>
<td>274,878 M</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2&lt;sup&gt;39&lt;/sup&gt;</td>
<td>549,756 M</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2&lt;sup&gt;40&lt;/sup&gt;</td>
<td>1,099,512 M</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2&lt;sup&gt;41&lt;/sup&gt;</td>
<td>2,199,024 M</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>2&lt;sup&gt;42&lt;/sup&gt;</td>
<td>4,398,048 M</td>
</tr>
<tr>
<td></td>
<td>64 (default)</td>
<td>2&lt;sup&gt;43&lt;/sup&gt;</td>
<td>8,796,096 M</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>2&lt;sup&gt;44&lt;/sup&gt;</td>
<td>17,592,192 M</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>2&lt;sup&gt;45&lt;/sup&gt;</td>
<td>35,184,384 M</td>
</tr>
<tr>
<td>4096 bytes (4K)</td>
<td>1</td>
<td>2&lt;sup&gt;38&lt;/sup&gt;</td>
<td>274,878 M</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2&lt;sup&gt;39&lt;/sup&gt;</td>
<td>549,756 M</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>2&lt;sup&gt;40&lt;/sup&gt;</td>
<td>1,099,512 M</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>2&lt;sup&gt;41&lt;/sup&gt;</td>
<td>2,199,024 M</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>2&lt;sup&gt;42&lt;/sup&gt;</td>
<td>4,398,048 M</td>
</tr>
<tr>
<td></td>
<td>32 (default)</td>
<td>2&lt;sup&gt;43&lt;/sup&gt;</td>
<td>8,796,096 M</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>2&lt;sup&gt;44&lt;/sup&gt;</td>
<td>17,592,192 M</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>2&lt;sup&gt;45&lt;/sup&gt;</td>
<td>35,184,384 M</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>2&lt;sup&gt;46&lt;/sup&gt;</td>
<td>70,368,768 M</td>
</tr>
<tr>
<td>Database block size</td>
<td>Records per block</td>
<td>Records per area</td>
<td>Approximate maximum records per area (in M)¹</td>
</tr>
<tr>
<td>---------------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>-------------------------------------------</td>
</tr>
<tr>
<td>2048 bytes (2K)</td>
<td>1</td>
<td>$2^{39}$</td>
<td>549,756 M</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$2^{40}$</td>
<td>1,099,512 M</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$2^{41}$</td>
<td>2,199,024 M</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>$2^{42}$</td>
<td>4,398,048 M</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>$2^{43}$</td>
<td>8,796,096 M</td>
</tr>
<tr>
<td></td>
<td>32 (default)</td>
<td>$2^{44}$</td>
<td>17,592,192 M</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>$2^{45}$</td>
<td>35,184,384 M</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>$2^{46}$</td>
<td>70,368,768 M</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>$2^{47}$</td>
<td>140,737,536 M</td>
</tr>
<tr>
<td>1024 bytes (1K)</td>
<td>1</td>
<td>$2^{40}$</td>
<td>1,099,512 M</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>$2^{41}$</td>
<td>2,199,024 M</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$2^{42}$</td>
<td>4,398,048 M</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>$2^{43}$</td>
<td>8,796,096 M</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>$2^{44}$</td>
<td>17,592,192 M</td>
</tr>
<tr>
<td></td>
<td>32 (default)</td>
<td>$2^{45}$</td>
<td>35,184,384 M</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>$2^{46}$</td>
<td>70,368,768 M</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>$2^{47}$</td>
<td>140,737,536 M</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>$2^{48}$</td>
<td>281,475,072 M</td>
</tr>
</tbody>
</table>

¹ 1 M = 1 million or 1,000,000
Table 2–3 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 areas are governed by addressability of 32-bit database keys.

<table>
<thead>
<tr>
<th>Block size</th>
<th>Records per block</th>
<th>Maximum blocks per area</th>
<th>Maximum area size</th>
</tr>
</thead>
<tbody>
<tr>
<td>8192 bytes (8K)</td>
<td>1</td>
<td>2,147,483,647</td>
<td>16TB</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1,073,741,823</td>
<td>8TB</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>536,870,911</td>
<td>4TB</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>268,435,455</td>
<td>2TB</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>134,217,727</td>
<td>1TB</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>67,108,863</td>
<td>512GB</td>
</tr>
<tr>
<td></td>
<td>64 (default)</td>
<td>33,554,431</td>
<td>256GB</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>16,777,215</td>
<td>128GB</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>8,388,607</td>
<td>64GB</td>
</tr>
<tr>
<td>4096 bytes (4K)</td>
<td>1</td>
<td>2,147,483,647</td>
<td>8TB</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1,073,741,823</td>
<td>4TB</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>536,870,911</td>
<td>2TB</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>268,435,455</td>
<td>1TB</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>134,217,727</td>
<td>512GB</td>
</tr>
<tr>
<td></td>
<td>32 (default)</td>
<td>67,108,863</td>
<td>256GB</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>33,554,431</td>
<td>128GB</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>16,777,215</td>
<td>64GB</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>8,388,607</td>
<td>32GB</td>
</tr>
<tr>
<td>Block size</td>
<td>Records per block</td>
<td>Maximum blocks per area</td>
<td>Maximum area size</td>
</tr>
<tr>
<td>-------------</td>
<td>-------------------</td>
<td>-------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>2048 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(2K)</td>
<td>1</td>
<td>2,147,483,647</td>
<td>4TB</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1,073,741,823</td>
<td>2TB</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>53,687,0911</td>
<td>1TB</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>268,435,455</td>
<td>512GB</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>134,217,727</td>
<td>256GB</td>
</tr>
<tr>
<td></td>
<td>32 (default)</td>
<td>67,108,863</td>
<td>128GB</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>33,554,431</td>
<td>64GB</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>16,777,215</td>
<td>32GB</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>8,388,607</td>
<td>16GB</td>
</tr>
<tr>
<td>1024 bytes</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(1K)</td>
<td>1</td>
<td>2,147,483,647</td>
<td>2TB</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>1,073,741,823</td>
<td>1TB</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>536,870,911</td>
<td>512GB</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>268,435,455</td>
<td>256GB</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>134,217,727</td>
<td>128GB</td>
</tr>
<tr>
<td></td>
<td>32 (default)</td>
<td>67,108,863</td>
<td>64GB</td>
</tr>
<tr>
<td></td>
<td>64</td>
<td>33,554,431</td>
<td>32GB</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>16,777,215</td>
<td>16GB</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>8,388,607</td>
<td>8GB</td>
</tr>
</tbody>
</table>
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 Table 2–4.

Table 2–4: RDBMS reserved recovery space

<table>
<thead>
<tr>
<th>Block size</th>
<th>Records per block</th>
<th>Reserved recovery space</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024</td>
<td>1,2,4,8,16,32,64</td>
<td>5G</td>
</tr>
<tr>
<td></td>
<td>128,256</td>
<td>1G</td>
</tr>
<tr>
<td>2048</td>
<td>1,2,4,8,16,32,64</td>
<td>5G</td>
</tr>
<tr>
<td></td>
<td>128,256</td>
<td>1G</td>
</tr>
<tr>
<td>4096</td>
<td>1,2,4,8,16,32,64,128,256</td>
<td>5G</td>
</tr>
<tr>
<td>8192</td>
<td>1,2,4,8,16,32,64,128,256</td>
<td>5G</td>
</tr>
</tbody>
</table>

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. Each table and each index can be assigned to only one storage area. Therefore, the size of a table or index is limited to the size of the storage area in which it resides. The following limits exist:

- The maximum number of **tables** supported is 32,767, regardless of block size.
- Tables have a maximum number of **fields**: 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:
  - In Release 10.1B and forward, and for 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.
  - Databases with 1K and 2K block sizes adhere to the previous index entry size of approximately 200 characters.
  - Databases migrated to Release 10.1C adhere to the previous index entry size of approximately 200 characters unless explicitly enabled to accept larger index entries.

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

Enable your migrated database with PROUTIL ENABLELARGEKEYS. For more information, see the “PROUTIL ENABLELARGEKEYS qualifier” section on page 21–55.
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. Table 2–5 lists the number of sequences per database block size.

Table 2–5: Maximum number of sequences

<table>
<thead>
<tr>
<th>Database block size</th>
<th>Maximum number of sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1024 bytes (1K)</td>
<td>250</td>
</tr>
<tr>
<td>2048 bytes (2K)</td>
<td>500</td>
</tr>
<tr>
<td>4096 bytes (4K)</td>
<td>1000</td>
</tr>
<tr>
<td>8192 bytes (8K)</td>
<td>2000</td>
</tr>
</tbody>
</table>

Databases created with OpenEdge Release 10.1B and later have 64-bit sequences. Databases migrated from a previous release, can enable support for 64-bit sequences with the PROUTIL ENABLESEQ64 command. See the “PROUTIL ENABLESEQ64 qualifier” section on page 21–56 for details.

Existing sequences with the upper limit specified as the Unknown value (?) were bounded by the maximum of a signed 32-bit integer in prior releases. When 64-bit sequences are enabled, they 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. Table 2–6 lists the maximum BI area size by block size used.

Table 2–6: Maximum primary recovery (BI) area size

<table>
<thead>
<tr>
<th>Block size</th>
<th>Maximum BI area size</th>
</tr>
</thead>
<tbody>
<tr>
<td>16,384 bytes (16K)</td>
<td>32TB</td>
</tr>
<tr>
<td>8192 bytes (8K) (default)</td>
<td>16TB</td>
</tr>
<tr>
<td>4096 bytes (4K)</td>
<td>8TB</td>
</tr>
<tr>
<td>2048 bytes (2K)</td>
<td>4TB</td>
</tr>
<tr>
<td>1024 bytes (1K)</td>
<td>2TB</td>
</tr>
</tbody>
</table>
Maximum database size

The maximum size of an OpenEdge Release 10.1C and later 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:

\[
\text{max database size} = \# \text{ of areas} \times \text{maximum area size} \\
= 31,994 \times 1 \text{ petabyte} \\
= \sim 32,000 \text{ petabytes}
\]

In prior releases, the maximum size of the database was constrained by the “2-billion row limit” which represents the maximum number of rows addressable with a 32-bit ROWID. In Release 10.1B and forward, ROWIDs in Type II storage areas are 64-bit values. The number of rows addressable by a 64-bit ROWID in a single table correspondingly expands; the maximum number of rows is now governed by the maximum size of an area (see Table 2–2). This increase in addressable rows is supported in Type II storage areas only.
Number of connections per database

Table 2–7 lists the maximum number of connections per database.

**Table 2–7: Maximum number of connections per database**

<table>
<thead>
<tr>
<th>Database type</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-user</td>
<td>1</td>
</tr>
<tr>
<td>Multi-user</td>
<td>Maximum number is machine dependent: up to 10,000 unless constrained by semaphore limits, process limits, or machine performance.</td>
</tr>
</tbody>
</table>
Number of simultaneous transactions per database

Table 2–8 lists the maximum number of simultaneous transactions per database.

Table 2–8: Maximum number of simultaneous transactions

<table>
<thead>
<tr>
<th>Database type</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-user</td>
<td>1</td>
</tr>
<tr>
<td>Multi-user</td>
<td>1 per user (maximum users = 10,000)</td>
</tr>
</tbody>
</table>
OpenEdge RDBMS Limits

OpenEdge database name limits

Table 2–9 lists the database name limits for each operating system.

Table 2–9: Database name limits

<table>
<thead>
<tr>
<th>Name type</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database names</td>
<td>One to 11 characters, excluding the pathname. You cannot use a file extension.</td>
</tr>
<tr>
<td>Pathnames</td>
<td>One to 255 characters, including the database name.</td>
</tr>
</tbody>
</table>

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:

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

\[
H = \text{Static Handles} + (\# \text{ of .dn files}) + (\# \text{ of .bn files}) + (\# \text{ of .an files})
\]

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.

Notes: Actual maximum values are constrained by available system resources and operating system settings.

For more information on shared memory segments, see the “Shared memory allocation” section on page 14–27.

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 10 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.
Table 2–10 illustrates the OpenEdge SQL data types and value limits.

### Table 2–10: SQL data type limits

<table>
<thead>
<tr>
<th>SQL data type</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIGINT</td>
<td>–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807</td>
</tr>
<tr>
<td>BINARY</td>
<td>2000 bytes</td>
</tr>
<tr>
<td>BIT</td>
<td>0 or 1</td>
</tr>
<tr>
<td>CHAR</td>
<td>2000 characters</td>
</tr>
<tr>
<td>DATE</td>
<td>Year: 1 to 9999; Month: 1 through 12; Date: 1 through the last day of the month</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>Defined in terms of precision and scale; precision=number of digits; scale=number of digits to the right of the decimal point. <strong>Note:</strong> The scale cannot be greater than precision—precision is limited to 50; scale is limited to 10</td>
</tr>
<tr>
<td>DOUBLE PRECISION</td>
<td>2.2250738585072014E–308 through 1.7976931348623157E+308</td>
</tr>
<tr>
<td>FLOAT</td>
<td>2.2250738585072014E–308 through 1.7976931348623157E+308</td>
</tr>
<tr>
<td>INTEGER</td>
<td>–2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>NUMERIC</td>
<td>Defined in terms of precision and scale; precision=number of digits; scale=number of digits to the right of the decimal point. <strong>Note:</strong> The scale cannot be greater than precision—precision is limited to 50; scale is limited to 10</td>
</tr>
<tr>
<td>REAL</td>
<td>1.175494351E–38F to 3.402823466E+38F</td>
</tr>
<tr>
<td>SMALLINT</td>
<td>–32,768 to 32,767</td>
</tr>
<tr>
<td>TIME</td>
<td>00:00:00 to 23:59:59</td>
</tr>
<tr>
<td>TIMESTAMP</td>
<td>Combination of Date and Time limits</td>
</tr>
<tr>
<td>TINYINT</td>
<td>–128 to 127</td>
</tr>
<tr>
<td>VARBINARY</td>
<td>31,995 bytes.</td>
</tr>
<tr>
<td>VARCHAR</td>
<td>31,995</td>
</tr>
</tbody>
</table>
Table 2–11 lists ABL data types and value limits.

### Table 2–11:  ABL data type limits

<table>
<thead>
<tr>
<th>ABL data type</th>
<th>Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BLOB</td>
<td>1GB</td>
</tr>
<tr>
<td>CHARACTER</td>
<td>Constrained by record size. <strong>Note:</strong> 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.</td>
</tr>
<tr>
<td>CLOB</td>
<td>1GB</td>
</tr>
<tr>
<td>DATE</td>
<td>1/1/32768 B.C. to 12/31/32767 A.D.</td>
</tr>
<tr>
<td>DATE-TIME</td>
<td>Same limit as DATE. TIME limit is: 00:00:00 to 23:59:59</td>
</tr>
<tr>
<td>DATE-TIME-TZ</td>
<td>Same limit as DATE-TIME. Time zone (TZ) limit is: –14:00 to +14:00</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>50 digits total; 1 to 10 decimal places</td>
</tr>
<tr>
<td>INTEGER</td>
<td>–2,147,483,648 to 2,147,483,647</td>
</tr>
<tr>
<td>INT64</td>
<td>–9,223,372,036,854,775,808 to 9,223,372,036,854,775,807</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>TRUE/FALSE, YES/NO</td>
</tr>
</tbody>
</table>

**Notes:** 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.
Table 2–12 describes data types supported by ABL and their corresponding SQL data types.

**Table 2–12: ABL and SQL data type correspondence**

<table>
<thead>
<tr>
<th>ABL data type</th>
<th>OpenEdge SQL data type</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHARACTER</td>
<td>VARCHAR</td>
</tr>
<tr>
<td>DATE</td>
<td>DATE</td>
</tr>
<tr>
<td>DECIMAL</td>
<td>DECIMAL or NUMERIC</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
</tr>
<tr>
<td>INT64</td>
<td>BIGINT</td>
</tr>
<tr>
<td>LOGICAL</td>
<td>BIT</td>
</tr>
<tr>
<td>RAW</td>
<td>VARBINARY</td>
</tr>
<tr>
<td>RECID</td>
<td>INTEGER</td>
</tr>
<tr>
<td>DATE-TIME</td>
<td>TIMESTAMP</td>
</tr>
</tbody>
</table>
This chapter describes how to start up and shut down an OpenEdge database, as detailed in the following sections:

- Management tools
- Starting a server or broker
- Starting and stopping background writers
- Stopping a server or broker
Management tools

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

- OpenEdge Management
- OpenEdge Explorer
- Progress Explorer

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.

Progress Explorer

Progress Explorer is a part of the Unified Broker system administration framework that provides a consistent interface for managing all OpenEdge RDBMS products installed on your network, consisting of the following elements:

- **AdminServer** — Provides secure administrative access to OpenEdge server products
- **Progress Explorer** — A graphical user interface that provides an easy way for you to manage OpenEdge servers
- **Command-line configuration utilities** — Character versions of the Progress Explorer configuration tools

AdminServer

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 the Progress Explorer configuration tools or 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*. 
Progress Explorer

Progress Explorer is a graphical administration utility that runs in Windows platforms. To use the Progress Explorer configuration tools, you must first start Progress Explorer and connect to a running AdminServer. Explorer then presents you with a view of all the products to which the AdminServer grants you administrative access.

You can select an instance of each of the products displayed and manage its operation or modify its configuration. For example, you can do the following:

- Connect to an AdminServer
- Start, stop, and query the status of OpenEdge databases and associated server groups

Managing database configurations

The database configurations you create with Progress Explorer are saved in the conmgr.properties file. It stores the database, configuration, and server group properties. When you use either the Progress Explorer database configuration tool 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 Progress Explorer to create and edit database configurations.

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 that are part of Progress Explorer 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.
Using the DBMAN command-line utility

After you use OpenEdge Explorer or Progress Explorer 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, Progress Explorer, or use 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 the OpenEdge Explorer or Progress Explorer configuration tools, which save them in the *conmgr.properties* file.

- **parameters**
  
  Specifies the startup parameters for the broker/server. See Chapter 19, “Database Startup Parameters,” for a list of broker/server startup parameters.

For more information about the PROSERVE command see Chapter 18, “Startup and Shutdown Commands.”

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 Progress Explorer to look up the operating character set name in the database configuration properties and reset it appropriately.

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 Progress Explorer to look up the log character set (LogCharacterSet) name in the database configuration properties and reset it appropriately.

Otherwise, use the Log Character Set (-cp1og) international startup parameter with the PROSERVE administration utility. For example:

```
proserve db-name -cp1og character-set-name
```

For more information on character sets and character conversion, see Chapter 21, “PROUTIL Utility.”

**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 Progress Explorer 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 the “PROWDOG command” section on page 18–15.
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.*
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.
Starting Up and Shutting Down

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 Chapter 14, “Managing Performance,” 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 the “Using APWs to improve performance” section on page 14–11.

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

To start an APW process, use OpenEdge Explorer, Progress Explorer, 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 Chapter 18, “Startup and Shutdown Commands.”
Starting and stopping background writers

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, Progress Explorer, 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, Progress Explorer, 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 Chapter 7, “After-imaging.”

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 the “PROSHUT command” section on page 3–12.

You can shut down a database using:

- OpenEdge Management or Explorer
- Progress 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 18–3 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.

Caution: Using -by with -F causes an emergency shutdown.

-Gw

For DataServers, specifies the DataServer broker to shut down.

For complete PROSHUT syntax, see the “PROSHUT command” section on page 18–11.

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

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Disconnect a User</td>
</tr>
<tr>
<td>2</td>
<td>Unconditional Shutdown</td>
</tr>
<tr>
<td>3</td>
<td>Emergency Shutdown (Kill All)</td>
</tr>
<tr>
<td>x</td>
<td>Exit</td>
</tr>
</tbody>
</table>
Table 3–1 lists the PROSHUT menu options and their actions.

### Table 3–1: PROSHUT menu options

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prompts you for the number of the user you want to disconnect.</td>
</tr>
<tr>
<td>2</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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 Table 3–2.

### Table 3–2: PROSHUT Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kill Users (-by)</td>
<td>Unconditional batch shutdown; kills all active users</td>
</tr>
<tr>
<td>Proceed If No Users (-bn)</td>
<td>Batch shutdown only if there are no active users</td>
</tr>
</tbody>
</table>

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 the “PROSHUT command” section on page 18–11.
**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 10
Database: /usr/WRK_DBS/101B/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

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 following figure shows an example of this option’s output:

```
Enter your selection:  8

<table>
<thead>
<tr>
<th>Usr</th>
<th>PID</th>
<th>Time of login</th>
<th>Userid</th>
<th>tty</th>
<th>Limbo?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6358</td>
<td>Dec 14 15:10:52</td>
<td>sue</td>
<td>/dev/ttyp0</td>
<td>no</td>
</tr>
<tr>
<td>4</td>
<td>7007</td>
<td>Dec 14 15:25:09</td>
<td>mary</td>
<td>/dev/ttyp5</td>
<td>no</td>
</tr>
</tbody>
</table>

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.
Part II

Protecting Your Data

Chapter 4, Backup Strategies
Chapter 5, Backing Up a Database
Chapter 6, Recovering a Database
Chapter 7, After-imaging
Chapter 8, Maintaining Security
Chapter 9, Auditing
Chapter 10, Transparent Data Encryption
Chapter 11, Replicating Data
Chapter 12, Failover Clusters
Chapter 13, Distributed Transaction Processing
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, as detailed in the following sections:

- 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, dn) and before-image files (.bn)**
  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 (.an)**
  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 (.tn)**
  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 4–1 contrasts OpenEdge and operating system backups.

Table 4–1: Backup options

<table>
<thead>
<tr>
<th>OpenEdge</th>
<th>Operating system</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online or offline</td>
<td>Offline only</td>
</tr>
<tr>
<td>Full or incremental</td>
<td>Full only</td>
</tr>
</tbody>
</table>

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. See the “Using database quiet points” section on page 5–9 for information on using quiet points in a backup. 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. Figure 4–1 illustrates how the size of an incremental backup varies, depending on the amount of changes to the database between incremental backups.

Figure 4–1:  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. Figure 4–2 illustrates how the Overlap qualifier determines the data backed up during an incremental backup.

For more information on the -io parameter, see the “PROBKUP utility” section on page 24–13.

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.

Figure 4–2: Incremental backup with overlap
## 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 Chapter 7, “After-imaging.”

- 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” section on page 7–8.

- 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. Table 4–2 lists questions to ask yourself when choosing backup media.

Table 4–2: Backup media questions

<table>
<thead>
<tr>
<th>To consider this . . .</th>
<th>Ask this question . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage capacity</td>
<td>Is the media large enough to contain all the files you need to back up?</td>
</tr>
<tr>
<td>Data transfer speed</td>
<td>Is the media fast enough to allow backup of files in a reasonable time?</td>
</tr>
<tr>
<td>Accessibility</td>
<td>Can you use the media at the time you need to run the backup?</td>
</tr>
<tr>
<td>Location</td>
<td>Is the device accessed through a network? If so, can the network handle the large volume of activity generated during a backup?</td>
</tr>
</tbody>
</table>
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**
- **Database size**
- **Time**
- **Unscheduled backups**

**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. See the “AI File Management utility” section on page 7–16 for more information.

**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
Back up a Database

Back 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. This chapter contains the following sections:

- 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:

Syntax

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

For more information on PROBKUP, see the “PROBKUP utility” section on page 24–13. For information on using PROBKUP to enable after-imaging and AI File Management, see Chapter 7, “After-imaging.”

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:

Syntax

```
probkup online db-name device-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 by entering the following command:

   ```
   proutil devel -C BUSY
   ```

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

   ```
   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.
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.

Table 5–1 lists the PROREST utility parameters you can use to test backups performed with PROBKUP.

Table 5–1: PROREST utility verification parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>-vp</td>
<td>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 the “CRC codes and redundancy in backup recovery” section on page 5–19 for more information about error correction blocks and data recovery. You can use the Partial Verify parameter with both online and offline backups.</td>
</tr>
<tr>
<td>-vf</td>
<td>Specifies that the PROREST utility will compare the backup block-for-block to the database.</td>
</tr>
</tbody>
</table>

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

For more information, see the “PROREST utility” section on page 24–28.

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 the “PROUTIL BUSY qualifier” section on page 21–19.

2. Shut down the database server.
   
   Before you back up the database, you must shut down the database server. See Chapter 3, “Starting Up and Shutting Down,” 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 ]
```

**Notes:** 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 Chapter 7, “After-imaging.”
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:

```
proquiet dbname enable
```

`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:

```
proquiet dbname disable
```

`dbname`

Specifies the name of the database for which you are disabling the database quiet processing point.

For more information and the complete syntax for PROQUIET, see Chapter 18, “Startup and Shutdown Commands.”

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 .st file updated in Step 4:

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

`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:

```
probkup dbname -norecover
```

`dbname`

Specifies the name 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.

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 Chapter 3, “Starting Up and Shutting Down,” for information about shutting down the database.

2. Verify that the database is not in use with the PROUTILITY BUSY utility, as shown:

   ```
   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 the “PROUTILITY BUSY qualifier” section on page 21–19.

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 the “Identifying files for backup” section on page 4–2 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 Chapter 23, “RFUTIL Utility.”
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 deve1.db database. To perform an online backup, skip Step 2 through Step 4.

To perform an incremental offline backup of the deve1.db database:

1. Prepare the backup media according to the operating system documentation.
2. Verify that the database is not in use by entering the following command:
   
   `proutil devel -C BUSY`

3. Shut down the deve1.db database by using the PROSHUT command:
   
   `proshut devel`

4. Run PROBKUP -estimate to determine how much media is necessary for the backup:
   
   `probkup devel incremental /dev/null -estimate`

5. Enter the following command to perform an incremental offline database backup:
   
   `probkup devel /dev/rrm/0m incremental -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 /dev/rrm/0m incremental -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.

`/dev/rrm/0m`

Specifies the output destination is a tape drive, `/dev/rrm/0m`.

**Caution:** If backing up to tape, do not use the same reel of tape that was used for the full backup.

`incremental`

Specifies that the backup is an incremental 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.
Database backup examples

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.

**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 by entering the following command:

```
proutil devel -C BUSY
```
3. Shut down `devel.db` by entering 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:

```
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. Enter the following command to perform a full offline 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
- Important rules for restoring backups
- Obtaining storage area descriptions using PROREST
- Database restore examples

Using the PROREST utility to restore a database

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

```
prorest dbname device-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 the “Obtaining storage area descriptions using PROREST” section on page 5–22.

`-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 the “Error-correction blocks” section on page 5–19 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 restore over an existing database, verify the tapes before doing the restore. If the existing database is the only copy, back up the existing database before doing the restore.

- Restore a backup with the same OpenEdge version that you used to perform the backup.

- You must restore an incremental database backup to an existing database.

- Create the 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. 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. 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. Enter the following command to run an incremental restore of the database from a tape once the full restore is done:

```
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`. 
This chapter explains the different ways to recover your OpenEdge database and transactions if your system or disks fail, as described in the following sections:

- 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. Figure 6–1 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.

![OpenEdge recovery mechanisms](image)

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:

```
11:13:54 Single-user session begin for marshall on /dev/pts/25 (451)
11:13:54 Begin Physical Redo Phase at 256 . (5326)
11:13:56 Physical Redo Phase Completed at blk 800 of 8165 and 31829 (7161)
11:13:56 Begin Physical Undo 1 transactions at blk 800 offset 8189 (7163)
11:14:38 Physical Undo Phase Completed at 1020 . (5331)
11:14:38 Begin Logical Undo Phase, 1 incomplete transactions are being backed out. (7162)
11:14:38 Logical Undo Phase Complete. (5329)
11:14:46 Single-user session end. (334)
```

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.
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.

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 Chapter 7, “After-imaging,” for information about enabling after-imaging.
- Perform regularly scheduled backups of the AI files. See Chapter 5, “Backing Up a Database,” for 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:

```plaintext
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 Chapter 13, “Distributed Transaction Processing.”

**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 the DB 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.

Figure 6–2 shows sample locations for each of the DB, BI, AI, and event log files.

Figure 6–2: 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.
Recovering a Database

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. Table 6–1 provides examples.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Recovery technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many committed transactions can you afford to lose?</td>
<td>1 week of transactions</td>
<td>Perform weekly backups.</td>
</tr>
<tr>
<td>How long can the application be offline while you perform scheduled maintenance?</td>
<td>8 hours per day</td>
<td>Perform weekly backups offline.</td>
</tr>
<tr>
<td>How long can you spend recovering if the database becomes unavailable?</td>
<td>24 hours</td>
<td>Keep a full backup and any number of incrementals or AI files.</td>
</tr>
<tr>
<td>Must distributed transactions occur consistently across databases?</td>
<td>No</td>
<td>Do not implement two-phase commit.</td>
</tr>
</tbody>
</table>

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. Table 6–2 provides examples.

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Recovery technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many committed transactions can you afford to lose?</td>
<td>1 day of transactions</td>
<td>Perform daily backups.</td>
</tr>
<tr>
<td>How long can the application be offline while you perform scheduled maintenance?</td>
<td>8 hours per week</td>
<td>Perform weekly offline backups with daily online full or incremental backups.</td>
</tr>
<tr>
<td>How long can you spend recovering if the database becomes unavailable?</td>
<td>8 hours</td>
<td>Keep a full backup and one incremental or AI file.</td>
</tr>
<tr>
<td>Must distributed transactions occur consistently across databases?</td>
<td>No</td>
<td>Do not implement two-phase commit.</td>
</tr>
</tbody>
</table>
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. Table 6–3 provides examples.

#### Table 6–3: Sample moderate-to-high availability requirements

<table>
<thead>
<tr>
<th>Question</th>
<th>Answer</th>
<th>Recovery technique</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many committed transactions can you afford to lose?</td>
<td>A few minutes of transactions</td>
<td>Enable after-imaging.</td>
</tr>
<tr>
<td>How long can the application be offline while you perform scheduled maintenance?</td>
<td>Never</td>
<td>Perform online backups with after-image files.</td>
</tr>
<tr>
<td>How long can you spend recovering if the database becomes unavailable?</td>
<td>24 hours</td>
<td>Restore a full backup and a series of any number of incrementals or AI files.</td>
</tr>
<tr>
<td>Must distributed transactions occur consistently across databases?</td>
<td>Yes</td>
<td>Implement two-phase commit.</td>
</tr>
</tbody>
</table>

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. Table 6–4 provides examples.

Table 6–4: Sample high availability requirements

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

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.

Figure 6–3 through Figure 6–7 show sample scenarios of a one-week period.

Figure 6–3: Recovery scenario—day 1

- Stop the database.
- Verify that the database is not in use.
- Back up the database files as a unit.
- Mark the database as backed up, unless you are using PROBKUP.
- Add after-image storage areas to the database and enable after-imaging.

Monday AM
- Start the database.

Monday PM
- Stop the database.
- Verify that the database is not in use.
- Back up the database files as a unit.
- Mark the database as backed up and establish a new busy AI file, unless you are using PROBKUP.
- Archive all the full AI files to media different than that used to back up the database.

Friday PM
- Start implementing after-imaging
- Verify that the database is not in use.
- Back up the database files as a unit.
- Mark the database as backed up, unless you are using PROBKUP.
- Add after-image storage areas to the database and enable after-imaging.
Figure 6–4:  Recovery scenario—day 2

- **Tuesday AM**
  - Start the database.

- **Tuesday Noon**
  - System failure
  - Restart the database. It automatically performs crash recovery.

- **Tuesday PM**
  - Perform a daily full backup
  - Stop the database.
  - Verify that the database is not in use.
  - Back up the database files as a unit.
  - Mark the database as backed up and establish a new busy AI file, unless you are using PROBKUP.
  - Archive all the full AI files to media different than that used to back up the database.
Figure 6–5: Recovery scenario—day 3

Wednesday AM
- Start the database.

Wednesday Noon
- Database disk lost
- Back up the busy and full AI files.
- Restore the last backup of database files. Do not restore any AI files.
- Roll forward the full and busy AI files.
- Back up the database files as a unit.
- Mark the database as backed up, unless you are using PROBKUP.
- Enable after-imaging.

Wednesday PM
- Perform a daily full backup
- Stop the database.
- Verify that the database is not in use.
- Back up the database files as a unit.
- Mark the database as backed up and establish a new busy AI file, unless you are using PROBKUP.
- Archive all the full AI files to media different than that used to back up the database.
Figure 6–6: Recovery scenario—day 4

Thursday AM
- Start the database.

Thursday PM
**Perform a full backup**
- Stop the database.
- Verify that the database is not in use.
- Back up the database files as a unit.
- Archive the AI file to a different type.
- Disable after-imaging.
- Run the batch job.

**Start a large batch job to run overnight**
Figure 6–7: Recovery scenario—day 5
### After-imaging and roll-forward recovery commands

Table 6–5 lists the utilities used to perform the steps in the scenarios shown in Figure 6–3 through Figure 6–7. These utilities are described in detail in Part IV, “Reference.”

#### Table 6–5: Utilities used with roll-forward recovery

<table>
<thead>
<tr>
<th>Utility command</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>Verifies that the database is not in use.</td>
</tr>
<tr>
<td></td>
<td>The database cannot be in use when you perform a full offline backup.</td>
</tr>
<tr>
<td>PROBKUP or operating system utility</td>
<td>Backs up the database.</td>
</tr>
<tr>
<td></td>
<td>You must perform a full backup of the database before you can use roll-forward recovery.</td>
</tr>
<tr>
<td>RFUTIL MARK BACKEDUP</td>
<td>Marks the database as backed up.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>RFUTIL AIMAGE NEW</td>
<td>Starts a new AI file.</td>
</tr>
<tr>
<td></td>
<td>After you back up the database and AI files, you can truncate the existing AI file.</td>
</tr>
<tr>
<td>RFUTIL AIMAGE END</td>
<td>Disables after-imaging.</td>
</tr>
<tr>
<td></td>
<td>Before you run a batch job, you might want to disable after-imaging to improve performance.</td>
</tr>
<tr>
<td>RFUTIL ROLL FORWARD RETRY</td>
<td>Re-starts the roll-forward operation.</td>
</tr>
<tr>
<td></td>
<td>Use of this qualifier is limited to RFUTIL roll-forward operations that fail because of power outages or system failures.</td>
</tr>
</tbody>
</table>
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:

```
rutil 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 AI file 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 Chapter 7, “After-imaging,” for more information.

5. Perform a full backup of the restored, recovered database. Follow the standard backup procedure described in Chapter 5, “Backing Up a Database.”

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 Chapter 7, “After-imaging,” 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 Chapter 5, “Backing Up a Database.”

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.

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.
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 the “PROUTIL TRUNCATE BI qualifier” section on page 21–105 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:

<table>
<thead>
<tr>
<th>ID</th>
<th>ShMemVer</th>
<th>Seg#</th>
<th>InUse</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>68</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(not OpenEdge)</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>0</td>
<td>Yes</td>
<td>/db/work5/sports</td>
</tr>
<tr>
<td>101</td>
<td>3</td>
<td>1</td>
<td>-</td>
<td>/db/work5/sports</td>
</tr>
<tr>
<td>120</td>
<td>3</td>
<td>0</td>
<td>No</td>
<td>/db/work5/test</td>
</tr>
<tr>
<td>150</td>
<td>2</td>
<td>-</td>
<td>-</td>
<td>(unsupported shared memory version)</td>
</tr>
</tbody>
</table>

Table 6–6 describes the display fields in the output.

**Table 6–6: Shared-memory segment status fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>Indicates the shared-memory ID.</td>
</tr>
<tr>
<td>ShMemVer</td>
<td>Specifies the shared-memory version.</td>
</tr>
<tr>
<td>Seg#</td>
<td>Indicates the shared-memory segment number. One database can own more than one segment.</td>
</tr>
<tr>
<td>InUse</td>
<td>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.</td>
</tr>
<tr>
<td>Database</td>
<td>Represents the full path name of the database.</td>
</tr>
</tbody>
</table>
Recovering from a lost or damaged control area

Use the PROSTRCT BUILDB 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 BUILDB 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 BUILDB assumes the name is `db-name.st`.

PROSTRCT BUILDB 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. Enter the following command:

   ```
   prostrct unlock db-name
   ```

   **Caution:** 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.

**p-dump.p**

```plaintext
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.
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 Chapter 16, “Dumping and Loading.”

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:

\[
100 + 32 \times \left(\frac{\text{database-size-in-bytes}}{\text{database-block-size}}\right)
\]

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. Run the PROUTIL command using the -F parameter:

   ```
   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 Chapter 16, “Dumping and Loading,” 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.
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. The following sections describe how to use after-imaging:

- 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. Figure 7–1 illustrates this behavior. An extent switch is the operation of switching from one AI area extent to another.

![Figure 7–1: After-image extents switching](image)

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 the “Monitoring AI file status” section on page 7–9.

Like database and BI extents, there are two types of AI extents:

- Fixed-length extents
- Variable-length extents
**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 AI data 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 Chapter 6, “Recovering a Database,” 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 the “After-image Stall (-aistall)” section on page 19–16.

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 the “Managing after-imaging files” section on page 7–9 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 Chapter 1, “Creating and Deleting Databases,” 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. Table 7–1 describes each token and value you enter for fixed-length files.

Table 7–1: File tokens

<table>
<thead>
<tr>
<th>Token</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage area type</td>
<td>Specifies the file type. Type a to indicate that this extent is an AI extent.</td>
</tr>
<tr>
<td>Extent path and file name</td>
<td>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.</td>
</tr>
<tr>
<td>Extent type</td>
<td>Specifies whether the extent is a fixed-length or variable-length extent. Type f for a fixed-length extent. It must be lowercase.</td>
</tr>
<tr>
<td>Extent size</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 Chapter 15, “Maintaining Database Structure.”

3. Create the initial database contents using the PROCOPY or PROREST utility. For more information about the PROCOPY and PROREST utilities, see Chapter 24, “Other Database Administration Utilities.”

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 7–1 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 Chapter 15, “Maintaining Database Structure.”

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 the “Creating after-image areas” section on page 7–5 for detailed instructions.

2. Back up your database.

3. After you back up your database, use the following command to enable after-imaging:

```
rfutil db-name -C aimage begin
```

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 the “Creating after-image areas” section on page 7–5 for detailed instructions.

2. Simultaneously backup your database online and enable after-imaging with PROBKUP, as shown:

   probkup online dbname output-device enableai

For complete PROBKUP syntax information, see the “PROBKUP utility” section on page 24–13.

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
- Switching to a new AI file
- Archiving an AI file
- Making an AI file available for reuse

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 the “AI File Management utility” section on page 7–16. 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 the description of the AIMAGE EXTENT LIST utility in Chapter 23, “RFUTIL Utility.”
- **PROMON**

Use **PROMON → R&D → Status Displays → AI Extents** to display information about each extent, including status. For example:

<table>
<thead>
<tr>
<th>Area</th>
<th>Status</th>
<th>Type</th>
<th>File Number</th>
<th>Size (KBytes)</th>
<th>Extent Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>BUSY</td>
<td>Fix</td>
<td>1</td>
<td>505</td>
<td>/usr1/101A/docsample.a1</td>
</tr>
<tr>
<td>14</td>
<td>EMPTY</td>
<td>Fix</td>
<td>0</td>
<td>505</td>
<td>/usr1/101A/docsample.a2</td>
</tr>
<tr>
<td>15</td>
<td>EMPTY</td>
<td>Var</td>
<td>0</td>
<td>121</td>
<td>/usr1/101A/docsample.a3</td>
</tr>
</tbody>
</table>

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
```
Managing after-imaging files

For more complete syntax details, including the `query-option` and `search-option` values, see the “RFUTIL AIMAGE QUERY qualifier” section on page 23–19.

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” section on page 23–15.

**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 the “PROUTIL ENABLELARGEFILES qualifier” section on page 21–54.

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 Chapter 19, “Database Startup Parameters.” 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.

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 the “Archiving an AI file” section on page 7–13 for details.

2. Use the RFUTIL AIMAGE NEW utility, as shown:

```
rutil 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 Chapter 23, “RFUTIL Utility.”
Archiving an AI file

Backing up the AI file involves:

- Scheduling backups
- Performing the backup
- Protecting the backup

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 Chapter 6, “Recovering a Database.” 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 the “RFUTIL AIMAGE EXTRACT qualifier” section on page 23–17.

**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.
Figure 7–2 shows the state transitions graphically.

![State transition diagram]

**Figure 7–2: 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. Normal database activity causes the current BUSY extent to fill. The extent is marked LOCKED, and the next EMPTY extent is marked as BUSY.
3. When OpenEdge Replication has replicated the LOCKED extent, it will mark the extent EMPTY.
4. 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 the “Enabling after-imaging offline” section on page 7–7. The example uses a database named mydb.

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 the “Enabling after-imaging offline” section on page 7–7 for instructions.
2. Create directories on disks with sufficient space to archive your AI extents.
3. Enable your database for AI file management:

   rfutil mydb -C aiarchiver enable

4. Start your database broker with AI file management parameters:

   proserve mydb -S 6931
   -aiarcdir /usr1/aiarchives/mydb/,/usr2/aiarchives/mydb/
   -aiarcinterval 120
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 the “Enabling after-imaging online” section on page 7–8 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, as shown:

   probkup online mydb /usr4/backups/mydb.bck enableaiarchiver -aiarcdir
   /usr1/aiarchives/mydb/,/usr2/aiarchives/mydb/
   -aiarcinterval 120

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 -aiastall parameter is implicitly included.

Setting -aiarchinterval to 120 directs the daemon to waken and archive extents every 2 minutes. You can specify an interval as small as 1 minute, 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 -aiarchinterval, 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:

```
rutil 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:

```
rutil 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 Chapter 23, “RFUTIL Utility.”

### 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:

\[ \text{directory-spec-database.ai-sequence.extent-name} \]

For example, if the name of your archive file is \texttt{usr1-dbs-sales.03.sales.a3}, you can reconstruct the following information:

- The source database is \texttt{/usr1/dbs/sales.db}
- The AI sequence number is \texttt{03}
- The extent name is \texttt{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 \texttt{.db} file, and is named, \texttt{database-name.archival.log}. The log file is created the first time information is written to it. The information stored in the archival 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,
Table 7–2 describes the record types supported in the archival log file.

Table 7–2: Archival log record types

<table>
<thead>
<tr>
<th>Record type value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0255</td>
<td><strong>Version indicator</strong> — This record indicates the version of archival log records that follow.</td>
</tr>
<tr>
<td>0001</td>
<td><strong>Automatic after-image file archive</strong> — This record describes the archival of an after-image extent by the AI File Management utility.</td>
</tr>
<tr>
<td>0003</td>
<td><strong>Manual after-image file archive</strong> — This record describes the manual archival of an after image file by the RFUTIL AIARCHIVE EXTENT command.</td>
</tr>
<tr>
<td>0004</td>
<td><strong>Manual after-image file archive to the NULL device</strong> — This record describes the manual archival of an after image file by the RFUTIL AIARCHIVE EXTENT command to the NULL device.</td>
</tr>
<tr>
<td>0032</td>
<td><strong>Full backup</strong> — This record describes a FULL online or offline backup.</td>
</tr>
<tr>
<td>0033</td>
<td><strong>Incremental backup</strong> — This record describes an INCREMENTAL backup.</td>
</tr>
</tbody>
</table>

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`
Table 7–3 describes the fields in a version indicator record.

### Table 7–3: Format of an archival log startup line

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record type (0255)</td>
<td>Header line indicator.</td>
</tr>
<tr>
<td>date</td>
<td>Date the AI File Management utility started in YYYYMMDD format.</td>
</tr>
<tr>
<td>time</td>
<td>Time the AI File Management utility started in HHMMSS format.</td>
</tr>
<tr>
<td>version</td>
<td>Version of the archival log records that follow. This field facilitates the programmatic parsing of the archive log file.</td>
</tr>
</tbody>
</table>

**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
```

Table 7–4 describes each field of an AI extent archive record.

### Table 7–4: Format of an AI extent archive record in the archive log file (1 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Current field position</th>
<th>Prior field position</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record type</strong></td>
<td>AI extent archive record values are:</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• 0001 — Automated extent archival</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 0003 — Manual extent archival</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 0004 — Manual extent archival to a NULL device</td>
<td></td>
<td></td>
</tr>
<tr>
<td>database</td>
<td>Name of the database</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>date</td>
<td>Date of the backup in YYYYMMDD format</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>time</td>
<td>Time of the backup in HHMMSS format</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>aibegin-date</td>
<td>Date that after imaging was started for the database in YYYYMMDD format</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>aibegin-time</td>
<td>Date that after imaging was started for the database in HHMMSS format</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>buseq</td>
<td>Backup sequence number of the backup</td>
<td>Removed</td>
<td>7</td>
</tr>
</tbody>
</table>
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
```

Table 7–5 describes each field of a backup record.

Table 7–4:  Format of an AI extent archive record in the archive log file  (2 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Current field position</th>
<th>Prior field position¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>aiseq</td>
<td>AI extent sequence number, with leading zeros (number of AI extents since last AIMAGE BEGIN)</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>extent-name</td>
<td>Source name of the archived extent</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>target-directory</td>
<td>The directory where the extent is archived</td>
<td>Removed</td>
<td>10</td>
</tr>
<tr>
<td>target-extent-name</td>
<td>The file name of archived extent</td>
<td>9</td>
<td>11</td>
</tr>
</tbody>
</table>

1. Field record position for an AI extent archive record prior to OpenEdge Release 10.1C02.

Table 7–5:  Format of a backup record in the archive log file  (1 of 2)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Current field position</th>
<th>Prior field position¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>Record type</td>
<td>Backup indicator record values are:</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>• 0032 — Full backup</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 0033 — Incremental backup</td>
<td></td>
<td></td>
</tr>
<tr>
<td>database</td>
<td>Name of the database</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>date</td>
<td>Date the backup record was written to the archive log file in YYYYMMDD format</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>time</td>
<td>Time the backup record was written to the archive log file in HHMMSS format</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>online indicator</td>
<td>Online indicator; possible values are:</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>• 0 — offline</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• 1 — online</td>
<td></td>
<td></td>
</tr>
<tr>
<td>buseq</td>
<td>Backup sequence number of the backup</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>backup-date</td>
<td>Date that the backup was started for the database in YYYYMMDD format</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>
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 the “OpenEdge Structure Add utility” section on page 15–8 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 the “PROSTRCT LIST qualifier” section on page 22–11.

The following example illustrates the effect of PROSTRCT REORDER AI on your AI extents:

- **Figure 7–3** shows the original AI extent configuration. The database will shutdown when A1 fills because it cannot switch to the full A2 extent.

![Figure 7–3: Original database AI extents](image-url)
The user must first add additional AI extents as Figure 7–4 shows.

Figure 7–4: 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. Figure 7–5 shows the result of the PROSTRCT REORDER AI.

Figure 7–5: 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:

Syntax

```bash
rfutil db-name -C roll forward -a ai-name
```

For complete syntax information see the “RFUTIL ROLL FORWARD qualifier” section on page 23–26.

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 the “RFUTIL ROLL FORWARD qualifier” section on page 23–26.

For more information about the ROLL FORWARD qualifier, see Chapter 6, “Recovering a Database” and Chapter 23, “RFUTIL Utility.”
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:

```bash
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:

```bash
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 recoveries 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.
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 the “RFUTIL SEQUENCE qualifier” section on page 23–32 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:

```bash
# 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
# prostruct add production_db add.st

# truncate the database BI file and mark as backed up
# prooutil 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.
# prostruct repair /db_standby/production_db /db_standby/standby.st
prostruct 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:

   rfutil db-name -C aimage end

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.
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. The following sections explain how to maintain the security of your OpenEdge database:

• 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

For more information on the security features of OpenEdge, see OpenEdge Getting Started: Core Business Services.

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.
Establishing an OpenEdge user ID and password

Every OpenEdge database connection has an associated user ID. Each type of OpenEdge application security uses the user ID to determine the user's authority to access data.

At the first level, connection security, you decide whether to allow all users the authority to connect to a database or to establish a list of authorized user IDs. Each user ID can be further protected by a password to prevent unauthorized users from connecting to the database.

If you choose not to establish a list of valid user IDs, then the database inherits user IDs or user names from the operating system.

On UNIX, the database uses the operating system user ID. Note that the user IDs or user names on the operating system must adhere to OpenEdge rules on user IDs. For more information, see the “OpenEdge user ID” section on page 8–2.

If you choose to establish a list of valid user IDs with passwords, then you must enter the data into the user list, that you access through the Data Administration Security menu. The database engine stores the user list in the _User table, which is hidden from users. From then on, users must specify a user ID and password when connecting to the database. The database no longer inherits the user ID or user name from the operating system once you have established this list.

In an interactive session, the application can prompt the user for a user ID and password. In a batch session, the user can specify a user ID and password by using the User ID (-U) and Password (-P) startup parameters.

OpenEdge user ID

A user ID is a string of up to 12 characters associated with a particular OpenEdge database connection. User IDs can consist of any printable character or digit except the following: #, *, !, and @. User IDs are not case sensitive; they can be uppercase, lowercase, or any combination of these. A user ID can be blank, written as the string “ ”, but you cannot define it as such through the Data Dictionary.

OpenEdge password

A password is a string of up to 16 characters that is associated with a user ID. When you add the password to the user list, the password is encoded with the ENCODE function. Because ENCODE returns different values for uppercase and lowercase input, all OpenEdge passwords are case sensitive.
Validating an OpenEdge user ID and password

If you establish a list of valid user IDs, 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 SETUSERID function to check the user ID and password entered by the user. 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, SETUSERID exits the user from the database. If the user ID and password combination is valid for the database, then SETUSERID 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), then the user is assigned the blank user ID. While you cannot prevent users from connecting to the database with the blank user ID, you can prevent them from accessing data by establishing compile-time and run-time security.

For more information about compile- and run-time security, see OpenEdge Getting Started: ABL Essentials.
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 ABL (Advanced Business Language) tables only, SQL tables only, or both ABL and SQL tables.

ABL tables only

By default, no user ID or password is required. You can assign a Security Administrator role. 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.

SQL tables 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.

Both ABL and SQL tables

When you create an OpenEdge database using the PROCOPY or PRODB commands, your login ID is used to automatically designate you as DBA, but only if there are no predefined _User records or DBAs (other than “sysprogress”) in the source database.

Then, as creator of the new database, you can:

- Use the Data Dictionary to designate an ABL DBA user ID and use it to grant access privileges.

- 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.
Connection security

Connection security ensures that only authorized users connect to a database. To establish connection security, you must create a list of valid users, and appoint one or more security administrators. Once connection security is established, users who are not security administrators have limited access to the Security menu; 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 the “Starting database brokers for SSL connections” section on page 3–8.

Designating valid users

When first defining the user list, access is not restricted. However, once established, only security administrators can add users to the user list. For information of defining security administrators, see the “Designating a security administrator” section on page 8–6.

To add a user to the user list:

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.

**Caution:** 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.

4. Enter the user name. The User Name field allows you to keep track of the user 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.

5. Enter the password and click OK. You are prompted you to verify the password.
6. Enter the password again. If you successfully enter the same password, you will see the user record added to the user list. If you enter a different password, no user record is created.

7. 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 list, 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 the “Changing a password” section on page 8–8 for instructions.

If users forget their passwords, you can delete their records from the user list and re-create new user records for them.

### Designating a security administrator

Once you establish a list of valid user IDs, you must 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 security administrator. OpenEdge allows you to designate security administrators only if you have connected to the database with a nonblank user ID and if your user ID is already in the user list. You can then designate yourself and other users as security administrators.

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**

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.

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.
To enter user IDs for security administrators:


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

Only security administrators can delete users from the user list.

To delete a user from the user list:

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.

   If you delete all user records from the user list, 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 list before you can exit to the main window.
Maintaining Security

Changing a password

Users can change their own passwords; they do not need security administrator privileges.

To change your password:

1. 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.

2. 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 or Data Administration tool 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 user 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.

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.
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. The following sections discuss auditing:

- Auditable events
- Auditing states
- Enabling and disabling auditing
- Auditing tables
- Archiving audit data
- Auditing impact on database resources
- Auditing impact on database utilities
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
- Security events
- Schema events
- Data events
- Administration events
- Utility events
- User events
- Application events

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
Auditable events

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 programatically 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.
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 earlier releases (10.0B and prior) **cannot** start or connect to the database.
  - Audit privileges of current (10.1A and forward) clients 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 earlier releases (10.0B and prior) **can** start and connect to the database.
  - Access to the audit data by current (10.1A and forward) clients is **disallowed**.
  - Audit privileges of current (10.1A and forward) clients 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 earlier releases (10.0B and prior) **cannot** start or connect to the database.
  - Access to the audit data by current (10.1A and forward) clients is **disallowed**.
  - Audit privileges of current (10.1A and forward) clients 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 Release 10.1A or higher database, the auditing meta-schema tables for audit policies, events, and data are created for the database. See the “Auditing tables” section on page 9–10 for descriptions of the tables and indexes. Enabling a database that is migrated from a release prior to 10.1A also adds additional meta-schema tables that are included in 10.1A and higher by default.

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 Chapter 15, “Maintaining Database Structure.”

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)```
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 the “PROUTIL ENABLEAUDITING qualifier” section on page 21–47.

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.

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:

**Syntax**

```
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 DISABLEAUDITING. 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. Table 9–1 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.

Table 9–1: Auditing meta-schema tables

<table>
<thead>
<tr>
<th>Table name</th>
<th>Description</th>
<th>Archived</th>
</tr>
</thead>
<tbody>
<tr>
<td>_aud-audit-data</td>
<td>This table contains the audit data records. All events are stored in this table.</td>
<td>Yes</td>
</tr>
<tr>
<td>_aud-audit-data-value</td>
<td>This table is a child table of the _aud-audit-data table, and contains records for each field change data event.</td>
<td>Yes</td>
</tr>
<tr>
<td>_aud-audit-policy</td>
<td>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.</td>
<td>No</td>
</tr>
<tr>
<td>_aud-event</td>
<td>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.</td>
<td>Yes</td>
</tr>
<tr>
<td>_aud-event-policy</td>
<td>This table contains policy settings for events associated with policies.</td>
<td>No</td>
</tr>
<tr>
<td>_aud-field-policy</td>
<td>This table contains field level auditing settings associated with a named policy.</td>
<td>No</td>
</tr>
<tr>
<td>_aud-file-policy</td>
<td>This table contains table level auditing settings associated with a named policy.</td>
<td>No</td>
</tr>
</tbody>
</table>
**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. Table 9–2 lists the indexes for each of the auditing-related schema tables. See *OpenEdge Getting Started: Core Business Services* for details on the components of each index.

**Table 9–2: Indexes for auditing schema tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Index name</th>
<th>Index type</th>
</tr>
</thead>
<tbody>
<tr>
<td>_aud-audit-data</td>
<td>_AppContext-Id</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Audit-time</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Connection-id</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Data-guid</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique</td>
</tr>
<tr>
<td></td>
<td>_Event-context</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Event-group</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_EventId</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_UserId</td>
<td>–</td>
</tr>
<tr>
<td>_aud-audit-data-value</td>
<td>_Field-name</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Continuation-seq</td>
<td>–</td>
</tr>
<tr>
<td>_aud-audit-policy</td>
<td>_Policy-active</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Policy-desc</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Policy-guid</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique</td>
</tr>
<tr>
<td></td>
<td>_Policy-name</td>
<td>Unique</td>
</tr>
<tr>
<td>_aud-event</td>
<td>_Event-desc</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Event-id</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td>_Event-name</td>
<td>Unique</td>
</tr>
<tr>
<td>_aud-event-policy</td>
<td>_Event-guid</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique</td>
</tr>
<tr>
<td>_aud-field-policy</td>
<td>_Field-name</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique</td>
</tr>
<tr>
<td></td>
<td>_File-field-owner</td>
<td>–</td>
</tr>
<tr>
<td>Table</td>
<td>Index name</td>
<td>Index type</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>_aud-file-policy</td>
<td>_Create-event-id</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Delete-event-id</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_File-owner</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Guid-file-owner</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique</td>
</tr>
<tr>
<td></td>
<td>_Read-event-id</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Update-event-id</td>
<td>–</td>
</tr>
<tr>
<td>_Client-session</td>
<td>_Auth-time</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Db-guid</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>_Session-uuid</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique</td>
</tr>
<tr>
<td></td>
<td>_Userid</td>
<td>–</td>
</tr>
<tr>
<td>_Db-detail</td>
<td>_Db-guid</td>
<td>Primary</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unique</td>
</tr>
</tbody>
</table>
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 Figure 9–1.

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 the “PROUTIL AUDITARCHIVE qualifier” section on page 21–12 and the “PROUTIL AUDITLOAD qualifier” section on page 21–15.

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.

Figure 9–1: Audit archive process

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 the “PROUTIL AUDITARCHIVE qualifier” section on page 21–12 and the “PROUTIL AUDITLOAD qualifier” section on page 21–15.
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 in Step 1 from the database, and does not produce a new archive file.
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 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 the “Specifying encrypted passwords” section on page 9–19.
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:

  ```bash
  $ proutil auditexampledb -C auditarchive -userid Mike -password guru
  ```

- Prompt for password:

  ```bash
  $ 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:

  ```bash
  $ 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 or the Data Administration online Help. For details on specifying encrypted passwords, see the “Specifying encrypted passwords” section on page 9–19.
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
  253e3b35331a20363202a330d353220325203536
  .
  .
  .
  proutil auditexampledb -C auditarchive -userid sysdba
  -password oech1::253e3b35331a20363202a330d353220325203536
  ```

For this example, assume that the DB Pass Key is “ultra_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. Table 9–3 details the components of an encrypted password.

Table 9–3: Encrypted password components

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oec</td>
<td>Abbreviation identifying the encryption algorithm, OpenEdge crypto</td>
</tr>
<tr>
<td>h1</td>
<td>A hex-binary encoded value identifying the encryption algorithm key-size as 1</td>
</tr>
<tr>
<td>::</td>
<td>Separator</td>
</tr>
<tr>
<td>password</td>
<td>Encrypted password; the encryption is performed with the algorithm specified in the prefix</td>
</tr>
</tbody>
</table>

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
- Protected areas
- Enhanced utilities
- Expanded PROUTIL syntax
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. Table 9–4 lists the protected tables.

Table 9–4: Protected tables

<table>
<thead>
<tr>
<th>Table name</th>
<th>Table type</th>
</tr>
</thead>
<tbody>
<tr>
<td>_aud-audit-data</td>
<td>Data</td>
</tr>
<tr>
<td>_aud-audit-data-value</td>
<td>Data</td>
</tr>
<tr>
<td>_client-session</td>
<td>Data</td>
</tr>
<tr>
<td>_aud-event</td>
<td>Policy</td>
</tr>
<tr>
<td>_aud-event-policy</td>
<td>Policy</td>
</tr>
<tr>
<td>_aud-audit-policy</td>
<td>Policy</td>
</tr>
<tr>
<td>_aud-file-policy</td>
<td>Policy</td>
</tr>
<tr>
<td>_aud-field-policy</td>
<td>Policy</td>
</tr>
</tbody>
</table>

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. Table 9–5 lists the protected utilities. The table also details the impacted objects and required privilege if applicable, and the access restrictions.

Table 9–5: Protected utilities

<table>
<thead>
<tr>
<th>Operation</th>
<th>PROUTIL qualifier</th>
<th>Impacted protected object</th>
<th>Required privilege</th>
<th>Restriction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bulk load</td>
<td>bulkload</td>
<td>Protected tables</td>
<td>–</td>
<td>Denied to all users</td>
</tr>
<tr>
<td>Binary load</td>
<td>load</td>
<td>Protected tables</td>
<td>–</td>
<td>Denied to all users</td>
</tr>
<tr>
<td>Binary dump</td>
<td>dump</td>
<td>dumpspecified</td>
<td>Protected tables</td>
<td>–</td>
</tr>
<tr>
<td>Index fix (delete record)</td>
<td>idxfix</td>
<td>Protected tables</td>
<td>Audit Archiver</td>
<td>Denied to all except Audit Archiver</td>
</tr>
</tbody>
</table>
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 ]] 
```
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. The following sections describe Transparent Database Encryption concepts and procedures:

- 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.

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. For a list of the ciphers supported for Transparent Data Encryption, see the “OpenEdge supported ciphers” section on page 10–6.
• 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*.

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 the “Running with Transparent Data Encryption enabled” section on page 10–16 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

**Notes:** 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 Chapter 21, “PROUTIL Utility.”

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 Table 10–1.

Table 10–1: Passphrase constraints

<table>
<thead>
<tr>
<th>Rule</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum number of characters</td>
<td>8</td>
</tr>
<tr>
<td>Maximum number of characters</td>
<td>2048</td>
</tr>
<tr>
<td>Minimum number of numeric characters</td>
<td>1</td>
</tr>
<tr>
<td>Minimum number of alpha characters</td>
<td>2</td>
</tr>
<tr>
<td>Minimum number of punctuation characters</td>
<td>1</td>
</tr>
<tr>
<td>Character set</td>
<td>[a-zA-Z0-9!@#$%^&amp;*()_+-{}</td>
</tr>
<tr>
<td>First character</td>
<td>(see Character set)</td>
</tr>
<tr>
<td>Mixed case alpha required</td>
<td>True</td>
</tr>
<tr>
<td>Case sensitive</td>
<td>True</td>
</tr>
</tbody>
</table>

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 Table 10–2 and Table 10–3.

### Table 10–2: Database Master key ciphers

<table>
<thead>
<tr>
<th>ID</th>
<th>Cipher</th>
<th>Mode</th>
<th>Size</th>
<th>Key type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AES</td>
<td>CBC</td>
<td>128</td>
<td>binary</td>
</tr>
<tr>
<td>2</td>
<td>AES</td>
<td>CBC</td>
<td>192</td>
<td>binary</td>
</tr>
<tr>
<td>3</td>
<td>AES</td>
<td>CBC</td>
<td>256</td>
<td>binary</td>
</tr>
<tr>
<td>4</td>
<td>DES</td>
<td>CBC</td>
<td>56</td>
<td>binary</td>
</tr>
<tr>
<td>5</td>
<td>DES3</td>
<td>CBC</td>
<td>168</td>
<td>binary</td>
</tr>
<tr>
<td>6</td>
<td>DES</td>
<td>CBC</td>
<td>56</td>
<td>PBE</td>
</tr>
<tr>
<td>7</td>
<td>RC4</td>
<td>ECB</td>
<td>128</td>
<td>binary</td>
</tr>
</tbody>
</table>

### Table 10–3: Object ciphers

<table>
<thead>
<tr>
<th>ID</th>
<th>Cipher</th>
<th>Mode</th>
<th>Size</th>
<th>Key type</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>NULL</td>
<td>NULL</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>1</td>
<td>AES</td>
<td>CBC</td>
<td>128</td>
<td>binary</td>
</tr>
<tr>
<td>2</td>
<td>AES</td>
<td>CBC</td>
<td>192</td>
<td>binary</td>
</tr>
<tr>
<td>3</td>
<td>AES</td>
<td>CBC</td>
<td>256</td>
<td>binary</td>
</tr>
<tr>
<td>4</td>
<td>DES</td>
<td>CBC</td>
<td>56</td>
<td>binary</td>
</tr>
<tr>
<td>5</td>
<td>DES3</td>
<td>CBC</td>
<td>168</td>
<td>binary</td>
</tr>
<tr>
<td>7</td>
<td>RC4</td>
<td>ECB</td>
<td>128</td>
<td>binary</td>
</tr>
</tbody>
</table>
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
- Running ENABLEENCRYPTION

Additional, optional steps, discussed in the following sections, include:

- Enabling BI file encryption after enabling encryption
- Enabling AI file encryption after enabling encryption
- Verifying enablements

**Note:** If you are enabling encryption on a database that is also configured for OpenEdge Replication, see the “Replication support” section on page 10–26 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 ENABLEENCRIPTION

One command enables your database for transparent data encryption. The basic syntax for enabling encryption is:

**Syntax**

```
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).
Enabling Encryption

- 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 the “Enabling BI file encryption after enabling encryption” section on page 10–9 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 the “PROUTIL ENABLEENCRYPTION” section on page 21–50 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 ENABLEENCRIPTION 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 dbname backupname -enableai -aiencryption enable
  ```

By default, if your database is enabled for encryption, enabling after-imaging enables encryption for your AI files.

**Notes:** 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
Figure 10–1 shows the output of PROUTIL DESCRIBE.

<table>
<thead>
<tr>
<th>OpenEdge Database Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Name</td>
</tr>
<tr>
<td>Version</td>
</tr>
<tr>
<td>Block Size</td>
</tr>
<tr>
<td>Largest Cluster</td>
</tr>
<tr>
<td>Create Date</td>
</tr>
<tr>
<td>Last Open Date</td>
</tr>
<tr>
<td>Prior Open Date</td>
</tr>
<tr>
<td>Schema Change Date</td>
</tr>
<tr>
<td>Before Imaging information</td>
</tr>
<tr>
<td>Block Size</td>
</tr>
<tr>
<td>Cluster Size (16K Units)</td>
</tr>
<tr>
<td>Last Open Date</td>
</tr>
<tr>
<td>Bi Encryption</td>
</tr>
<tr>
<td>After Imaging Information</td>
</tr>
<tr>
<td>Block Size</td>
</tr>
<tr>
<td>Begin Date</td>
</tr>
<tr>
<td>Last AIMAGE NEW</td>
</tr>
<tr>
<td>Currently Busy Area</td>
</tr>
<tr>
<td>Current File Number</td>
</tr>
<tr>
<td>Ai Encryption</td>
</tr>
<tr>
<td>Backup Information</td>
</tr>
<tr>
<td>Last Full Backup Date</td>
</tr>
<tr>
<td>Last Incremental Backup</td>
</tr>
<tr>
<td>Database Features</td>
</tr>
<tr>
<td>ID   Feature</td>
</tr>
<tr>
<td>9  64 Bit DBKEYS</td>
</tr>
<tr>
<td>10 Large Keys</td>
</tr>
<tr>
<td>11 64 Bit Sequences</td>
</tr>
<tr>
<td>13 Encryption</td>
</tr>
</tbody>
</table>

Figure 10–1: Output of PROUTIL DESCRIBE

For more information on PROUTIL DESCRIBE, see the “PROUTIL DESCRIBE qualifier” section on page 21–30.
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 the “Encrypting your existing data” section on page 10–14 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 the “Creating encryption policies with PROUTIL EPOLICY” section on page 10–12.
- To create a policy with Data Admin, see the “Creating encryption policies with Data Admin” section on page 10–13.
- To create a policy with OpenEdge SQL, see the “Encrypting your existing data” section on page 10–14.

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:

**Syntax**

```
proutil db-name -C epolicy manage object-type encrypt object-name
```

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"
```

Note that EPOLICY MANAGE operations on areas must be performed while the database is offline. See the “PROUTIL EPOLICY MANAGE qualifier” section on page 21–59 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 the “Creating encryption policies with PROUTIL EPOLICY” section on page 10–12 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 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
- Autostart
- Temporary files
- Restricted utility access
- Database backups
- Moving tables and indexes
- Binary dump and load of data

**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.

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, as shown:

```
proenv>proutil t1demo -C epolicy manage keystore userphrase -Passphrase
```

2. Change your autostart configuration by specifying key store user account:

```
proenv>proutil t1demo -C epolicy manage autostart user -Passphrase
```

This command modifies encryption access control in the Keystore file. After successful completion of the command, the Keystore file must be backed-up. (15518)

Please Retype your Passphrase for Verification
Enter new passphrase [required] :

proenv>

proenv>
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 ENABLEENCRIPTION
- PROUTIL DISABLEENCRYPTION
- PROUTIL TABLEMOVE
- PROUTIL IDXMIME
- 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. See the “PROUTIL EPOLICY MANAGE qualifier” section on page 21–59 for the complete syntax.

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. Table 10–4 describes the results of moving objects (tables or indexes) in an encryption-enabled database.

Table 10–4: Encryption impact of moving tables and indexes

<table>
<thead>
<tr>
<th>If you are moving . . .</th>
<th>To . . .</th>
<th>Then . . .</th>
</tr>
</thead>
</table>
| 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 |
<table>
<thead>
<tr>
<th>If you are moving . . .</th>
<th>To . . .</th>
<th>Then . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>A table or index from an encrypted Type I area</td>
<td>An unencrypted Type I area</td>
<td>• Execution of this operation is restricted to an authenticated DBA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• No change in encryption policy (the source policy is for the entire area)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Object is unencrypted in the destination area</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• User is prompted to confirm the move and the resulting decryption of the object</td>
</tr>
<tr>
<td>An encrypted Type I area</td>
<td></td>
<td>• Execution of this operation is restricted to an authenticated DBA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• No change in encryption policy (the source and target policies are for the entire area)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Object is encrypted in the destination area with encryption policy of the destination area</td>
</tr>
<tr>
<td>A Type II area</td>
<td></td>
<td>• Execution of this operation is restricted to an authenticated DBA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A new encryption policy is created for the object (with the same cipher as the source area policy)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Object is encrypted in the destination area</td>
</tr>
</tbody>
</table>
Table 10–4: Encryption impact of moving tables and indexes

<table>
<thead>
<tr>
<th>If you are moving . . .</th>
<th>To . . .</th>
<th>Then . . .</th>
</tr>
</thead>
</table>
| 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 |
| 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 the following case-sensitive qualifier to the command line:

```
-Cipher 6
```

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 `-Cipher 6` to the command line to indicate that you are loading encrypted data. You will be prompted to enter and verify the passphrase that you entered when you created the dump.

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*.

Table 10–5 lists the audit events related to Transparent Data Encryption.

**Table 10–5: Auditing events for Transparent Data Encryption (1 of 4)**

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Event Name</th>
<th>Detects</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11000</td>
<td>_sys.db.enc.enable</td>
<td>Enable encryption</td>
<td>PROUTIL ENABLEENCRIPTION</td>
</tr>
<tr>
<td>11001</td>
<td>_sys.db.enc.disable</td>
<td>Disable encryption</td>
<td>PROUTIL DISABLEENCRIPTION</td>
</tr>
<tr>
<td>11100</td>
<td>_sys.ks.create</td>
<td>Create a new OpenEdge key store</td>
<td>PROUTIL ENABLEENCRIPTION; PROUTIL EPOLICY MANAGE key store reconstruct</td>
</tr>
<tr>
<td>11101</td>
<td>_sys.ks.delete</td>
<td>Delete an existing OpenEdge key store</td>
<td>PROUTIL DISABLEENCRIPTION</td>
</tr>
</tbody>
</table>
Table 10–5: Auditing events for Transparent Data Encryption

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Event Name</th>
<th>Detects</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11102</td>
<td>_sys.ks.open.pass</td>
<td>Successfully opened an OpenEdge key store</td>
<td>PROUTIL EPOLICY MANAGE; internal ABL and SQL commands</td>
</tr>
<tr>
<td>11103</td>
<td>_sys.ks.rekey</td>
<td>Changed OpenEdge key store’s encryption key</td>
<td>Not in use for this release</td>
</tr>
<tr>
<td>11104</td>
<td>_sys.ks.setcipher</td>
<td>Changed OpenEdge key store encryption cipher</td>
<td>Not in use for this release</td>
</tr>
<tr>
<td>11105</td>
<td>_sys.ks.setadmin.pwd</td>
<td>Changed OpenEdge key store’s admin passphrase</td>
<td>PROUTIL EPOLICY MANAGE key store adminphrase …</td>
</tr>
<tr>
<td>11106</td>
<td>_sys.ks.setuser.pwd</td>
<td>Changed OpenEdge key store’s user passphrase</td>
<td>PROUTIL EPOLICY MANAGE key store userphrase …</td>
</tr>
<tr>
<td>11107</td>
<td>_sys.ks.ke.create.pass</td>
<td>Successfully created new encryption key entry</td>
<td>PROUTIL ENABLEENCRYPTION; PROUTIL EPOLICY MANAGE key store reconstruc…</td>
</tr>
<tr>
<td></td>
<td>_sys.ks.ke.update.pass</td>
<td>Successfully changed a key entry’s owner, passphrase or state</td>
<td>PROUTIL DISABLEENCRYPTION</td>
</tr>
<tr>
<td>11109</td>
<td>_sys.ks.ke.delete.pass</td>
<td>Successfully deleted a key entry</td>
<td>Not in use for this release</td>
</tr>
<tr>
<td>11110</td>
<td>_sys.ks.ke.read.pass</td>
<td>Successfully returned a clear-text encryption key</td>
<td>_mprosrv; database utilities; ABL single-user database connection</td>
</tr>
<tr>
<td>11111</td>
<td>_sys.ks.open.fail</td>
<td>Failed when attempting to open a OpenEdge key store</td>
<td>_mprosrv; database utilities; ABL single-user database connection</td>
</tr>
<tr>
<td>11112</td>
<td>_sys.ks.ke.create.fail</td>
<td>Failed when attempting to create a new key entry</td>
<td>PROUTIL ENABLEENCRYPTION</td>
</tr>
<tr>
<td>11113</td>
<td>_sys.ks.ke.update.fail</td>
<td>Failed when attempting to change a key entry’s owner, passphrase, or state</td>
<td>PROUTIL DISABLEENCRYPTION</td>
</tr>
<tr>
<td>11114</td>
<td>_sys.ks.ke.delete.fail</td>
<td>Failed when attempting to delete a key entry</td>
<td>Not in use for this release</td>
</tr>
<tr>
<td>11200</td>
<td>_sys.as.create.pass</td>
<td>Successfully created new autostart credentials (for key store access)</td>
<td>PROUTIL ENABLEENCRYPTION</td>
</tr>
<tr>
<td>11201</td>
<td>_sys.as.delete.pass</td>
<td>Successfully deleted existing autostart credentials (for key store access)</td>
<td>PROUTIL DISABLEENCRYPTION</td>
</tr>
</tbody>
</table>
Table 10–5: Auditing events for Transparent Data Encryption

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Event Name</th>
<th>Detects</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11202</td>
<td>_sys.as.open.pass</td>
<td>Successfully opened autostart credentials and accessed the key store</td>
<td>_mprosrv; database utilities; ABL single-user database connection</td>
</tr>
<tr>
<td>11203</td>
<td>_sys.as.recover.pass</td>
<td>Successfully forced new autostart credentials and OpenEdge key store passphrase credentials after lockout condition entered</td>
<td>PROUTIL EPOLICY MANAGE key store reconstruct</td>
</tr>
<tr>
<td>11204</td>
<td>_sys.as.update.pass</td>
<td>Successfully updated autostart credentials (for key store access)</td>
<td>PROUTIL EPOLICY MANAGE KEYSTORE [autostart</td>
</tr>
<tr>
<td>11205</td>
<td>_sys.as.open.fail</td>
<td>Successfully opened autostart credentials but failed to access the key store</td>
<td>_mprosrv; database utilities; ABL single-user database connection</td>
</tr>
<tr>
<td>11206</td>
<td>_sys.as.update.fail</td>
<td>Failure found when updating autostart credentials</td>
<td>PROUTIL EPOLICY MANAGE KEYSTORE [autostart</td>
</tr>
<tr>
<td>11207</td>
<td>_sys.as.recover.fail</td>
<td>Failed to forced new autostart credentials and OpenEdge key store passphrase credentials after lockout condition entered</td>
<td>PROUTIL EPOLICY MANAGE key store reconstruct</td>
</tr>
<tr>
<td>11300</td>
<td>_sys.enc.scan</td>
<td>Started a scan of an encrypted object to determine the # of blocks related to each object security policy</td>
<td>PROUTIL EPOLICY SCAN</td>
</tr>
<tr>
<td>11301</td>
<td>_sys.enc.update</td>
<td>Started an update of an encrypted object to re-encrypt previous policy encrypted block with the current policy</td>
<td>PROUTIL EPOLICY UPDATE</td>
</tr>
<tr>
<td>11400</td>
<td>_sys.db.dbpolicy.create</td>
<td>Create a new version of a database master key’s security policy</td>
<td>PROUTIL ENABLEENCRYPTION</td>
</tr>
<tr>
<td>11401</td>
<td>_sys.db.dbpolicy.update</td>
<td>Updated an existing database master key’s security policy’s information</td>
<td>Not in use for this release</td>
</tr>
</tbody>
</table>
Table 10–5: Auditing events for Transparent Data Encryption

<table>
<thead>
<tr>
<th>Event ID</th>
<th>Event Name</th>
<th>Detects</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>11402</td>
<td>_sys.db.dbpolicy.delete</td>
<td>Deleted an existing database master key’s security policy [version]</td>
<td>PROUTIL DISABLEENCRYPTION</td>
</tr>
<tr>
<td>11500</td>
<td>_sys.db.objpolicy.create</td>
<td>Create a new version of a database object’s security policy</td>
<td>PROUTIL EPOLICY MANAGE; internal ABL and SQL commands</td>
</tr>
<tr>
<td>11501</td>
<td>_sys.db.objpolicy.update</td>
<td>Update an existing database object security policy’s state</td>
<td>PROUTIL EPOLICY MANAGE; internal ABL and SQL commands</td>
</tr>
<tr>
<td>11600</td>
<td>_sys.db.pwdpolicy.create</td>
<td>Create a new version of a database passphrase rules policy</td>
<td>PROUTIL ENABLEENCRYPTION</td>
</tr>
<tr>
<td>11601</td>
<td>_sys.db.pwdpolicy.update</td>
<td>Update an existing [version] of a database passphrase rules policy</td>
<td>Not in use for this release</td>
</tr>
<tr>
<td>11602</td>
<td>_sys.db.pwdpolicy.delete</td>
<td>Delete an existing [version] of a database passphrase rules policy</td>
<td>PROUTIL DISABLEENCRYPTION</td>
</tr>
</tbody>
</table>

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 the “Rekey encryption policies with Data Admin” section on page 10–27.
- To rekey a policy from the command line, see the “Rekey encryption policies with PROUTIL EPOLICY” section on page 10–28.
- To rekey a policy with OpenEdge SQL, see the “Encrypting your existing data” section on page 10–14.

You can also view the history of an encryption policy. See the “Viewing encryption policy history in Data Admin” section on page 10–29 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:

![Generate Encryption Keys dialog](image)

**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**:

![Confirmation dialog](image)

**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
```
Similarly, 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.


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.

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.
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:

Syntax

```
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.
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.

This chapter contains the following sections:

- Replication schemes
- Replication models
- 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 OpenEdge Getting Started: ABL Essentials 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 the “Implementing log-based site replication” section on page 11–6.

Replication models

At the highest level, there are two major models of replication: synchronous and asynchronous.

In a synchronous replication model, data replication occurs within the scope of the original transaction. In other words, replication occurs transaction by transaction. Typically, this model is implemented using a two-phase commit protocol. Two-phase commit ensures that distributed transactions occur consistently across databases. For more information, see Chapter 13, “Distributed Transaction Processing.”

Because the data modifications are replicated as part of the original transaction, synchronous replication ensures high data availability and consistency. The entire transaction is either committed to both systems or backed out completely.

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.

**Figure 11–1** illustrates the data distribution model.

![Data distribution model](image)

**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.
Figure 11–2 illustrates two data consolidation models, one with no data ownership, and the other with table partitioning.

**Figure 11–2:  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.

Figure 11–3 illustrates the 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 Chapter 7, “After-imaging,” and Chapter 5, “Backing Up a Database.”

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 Chapter 7, “After-imaging.”

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 Chapter 15, “Maintaining Database Structure.”

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 Chapter 5, “Backing Up a Database.”

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 Chapter 23, “RFUTIL Utility.”

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 Chapter 6, “Recovering a Database.”

Figure 11–4 shows after-image extent files replicated from the primary to a secondary database.
Failover Clusters

Failover Clusters provide an operating system and hardware vendor-independent solution for automated fail over of your OpenEdge database and related resources.

This chapter describes Failover Clusters in the following sections:

- Overview
- 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

Throughout the remainder of this chapter, Failover Clusters is also referred to as Clusters.
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 the “Related software and hardware” section on page 12–2 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.

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 (32-bit and 64-bit)**

IBM software:

- AIX5L V5.3
- HACMP 5.3 cluster manager
**Overview**

**HPUX (32-bit and 64-bit)**

HP software:
- HPUX 11.i or later
- HP mc/ServiceGuard 11.x or later

**HPUX (Itanium 2)**

HP software:
- HPUX 11.23 or later
- HP mc/ServiceGuard 11.x or later

**SUN Solaris Sparc (32-bit and 64-bit)**

SUN software:
- SUN Solaris Sparc 9
- SUN Cluster Version 3.0

**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, the OpenEdge Enterprise product must be installed 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 the updated and re-enabled afterwards.
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
- For HPUX 32-bit only, set `JDKHOME` in the default system profile

Set the `JDKHOME` environment variable in `/etc/profile` so that the Java environment can be configured to properly support SQL Java Stored Procedures and Java Triggers. `JDKHOME` should be set to point to the installation for the JDK.

**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 `$PSC_CLUSTER_PATH/bin`
- Modify `LD_LIBRARY_PATH` to include `$PSC_CLUSTER_PATH/lib` and `$PSC_CLUSTER_PATH/bin`
- Set `DLC` to the install directory for OpenEdge
Windows

- 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, so its rate of growth is very small and it can be deleted at the customer’s discretion.
Required cluster components

Figure 12–1 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.

![Cluster component configuration diagram]

Figure 12–1: 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. Figure 12–2 shows this relationship. The cluster manager software must know about OpenEdge to handle it properly in the event of a failover. See the “Using the PROCLUSTER command-line interface” section on page 12–11 for information on using PROCLUSTER to cluster-enable a database.

Figure 12–2: 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 12–1** 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. **Figure 12–3** 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 12–3:** 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 the “PROCLUSTER utility” section on page 24–17.

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`

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 the “Starting a cluster-enabled database” section on page 12–13.
- Cannot be deleted without first being disabled—see the “Disabling a cluster-enabled database” section on page 12–12.
- Cannot have its physical structure altered without being re-registered—see the “Changing the structure of the database” section on page 12–15.

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:

```
x.x.x.x prostrct cluster session begin for userid on CON:. (451)
x.x.x.x 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
```

Where 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.

To verify that the database started correctly, use the `isalive` and `looksalive` parameters to PROCLUSTER. See the “Isalive and looksalive” section on page 12–14 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 the “Isalive and looksalive” section on page 12–14.

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 only database state that returns a value of Looks Alive is 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 32 and 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 Chapter 22, “PROSTRCT Utility.”

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.
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

**Note:** Do not add extents to a volume group or file system already in use by another resource group.
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 32 bit and 64 bit

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 32 bit and 64 bit

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
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 Progress Explorer to indicate that it is in a clustered environment.

To start the AdminServer in a cluster, enter the following command:

```
proadsv -start -cluster -host cluster-alias
```

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.

To 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 Progress Explorer and connect to the AdminServer for the cluster.
3. **Right-click** to add a new database. Enter the database name and check the **Start this Database in Cluster mode** check box, as shown:

![Database Properties](image)

4. Complete the database configuration.
5. 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
```

6. Start the database using Progress 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 the “PROSERVE command” section on page 18–9 for a complete description of the PROSERVE command. See the “Cluster-enabling a database” section on page 12–11 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, Cluster Configuration, 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 Cluster Administrator.

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.
   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.

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.

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.

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.
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 the “Disabling a cluster-enabled database” section on page 12–12.
Unix cluster management commands

Use the commands listed in Table 12–1 to manage your cluster resources with your Unix cluster manager. For information on any of these commands, see your cluster manager documentation.

**Table 12–1: Unix cluster management commands**

<table>
<thead>
<tr>
<th>Generic function</th>
<th>AIX</th>
<th>HPUX</th>
<th>Solaris</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default script generation</td>
<td>n/a</td>
<td>cmmakepkg</td>
<td>n/a</td>
</tr>
<tr>
<td>Enumerate cluster objects</td>
<td>clfindres, clgetgrp, clgetaddr, cl1scf</td>
<td>cmquerycl, cmgetconf, cmscanc1, cmviewcl, cmviewconf</td>
<td>scha_cluster_open, scha_resource_open, scha_resourcetype_open</td>
</tr>
<tr>
<td>Validate configuration</td>
<td>clverify_cluster</td>
<td>cmcheckconf</td>
<td>n/a</td>
</tr>
<tr>
<td>View cluster or resource group status</td>
<td>clstat, clRGinfo</td>
<td>cmviewcl</td>
<td>scstat</td>
</tr>
<tr>
<td>Register resource</td>
<td>cl_crlvfs, cl_mkv, cl_mkgroup, cl_mklv, cl_updatevg</td>
<td>capplyconf</td>
<td>scha_control, scha_resource_setstatus</td>
</tr>
<tr>
<td>Remove resource</td>
<td>cl_rmfs, cl_rmgrou, cl_rmlv, cl_updatevg</td>
<td>cmdeleteconf</td>
<td>scha_control, scha_resource_setstatus</td>
</tr>
<tr>
<td>Start resource</td>
<td>n/a</td>
<td>cmrunpkg</td>
<td>scha_control, scha_resource_setstatus</td>
</tr>
<tr>
<td>Stop resource</td>
<td>n/a</td>
<td>cmhaltpkg</td>
<td>scha_control, scha_resource_setstatus</td>
</tr>
<tr>
<td>Move resource to node</td>
<td>n/a</td>
<td>cmhaltpkg, cmrunpkg</td>
<td>scha_control, scha_resource_setstatus</td>
</tr>
</tbody>
</table>
For AIX and HP, shared disks must be made available for use, varied-on, and mounted. Use the commands listed in Table 12–2 to vary-on shared resources. For information on any of these commands, see your cluster manager documentation.

**Table 12–2:  UNIX Shared disk commands**

<table>
<thead>
<tr>
<th>Generic function</th>
<th>AIX</th>
<th>HPUX</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vary-on volume group</td>
<td><code>varyonvg volgrp</code></td>
<td><code>vgchange -a y volgrp</code></td>
</tr>
<tr>
<td>Mount shared file system for general use</td>
<td><code>mount/sharedfs</code></td>
<td><code>mount/sharedfs</code></td>
</tr>
<tr>
<td>List volume groups</td>
<td><code>lsvg -o</code></td>
<td><code>vgdisplay</code></td>
</tr>
</tbody>
</table>
Distributed transactions involve two or more databases in a single transaction, as described in the following sections:

- 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 Figure 13–1.

![Diagram showing data inconsistency](image-url)

**Figure 13–1:** Data inconsistency
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.
Figure 13–2 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.

As Figure 13–2 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. Figure 13–3 illustrates a limbo transaction.

**Figure 13–3: 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 Chapter 6, “Recovering a Database,” 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 the “Transaction log area” section on page 13–8.

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:

**Syntax**

```bash
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:

```bash
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:

\[
\begin{align*}
1,000,000 \times 1.2 &= 1,200,000 \\
1,200,000 / 16384 &= 73.242 \text{ (rounded up to 74)} \\
74 \times 16384 &= 1212416 \\
1212416 / 1024 &= 1184 \text{ KB}
\end{align*}
\]

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:

\[
\text{proutil db-name -C 2phase end}
\]

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.
- Cannot roll back a limbo transaction.

Figure 13–4 illustrates where messages are displayed.

<table>
<thead>
<tr>
<th>During normal operations:</th>
<th>When limbo transactions occur:</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Remote client</strong>&lt;br&gt;Does not write to database.</td>
<td><strong>Remote client</strong>&lt;br&gt;Writes messages to screen.</td>
</tr>
<tr>
<td><strong>Self-service client</strong>&lt;br&gt;Writes to database.</td>
<td><strong>Self-service client</strong>&lt;br&gt;Writes messages to screen and log file.</td>
</tr>
<tr>
<td><strong>Server</strong>&lt;br&gt;Writes to database.</td>
<td><strong>Server</strong>&lt;br&gt;Writes messages to log file.</td>
</tr>
</tbody>
</table>

**Figure 13–4: 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:

   MONITOR Release 10.0A
   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

   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
   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  sports 1  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:

```
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  sports 1  42453
```

Write down this information. You will use it to resolve the limbo transaction.

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:

```plaintext
MONITOR Release 10
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 Step 3, 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:

```plaintext
  1  Abort a Limbo Transaction
  2  Commit a Limbo Transaction
  Q  Quit
Enter choice>
```

9. To commit the transaction, choose 2 (Commit a Limbo Transaction). PROMON prompts you to enter the user number you recorded in Step 3, then press RETURN. PROMON displays a message similar to the following:

```plaintext
User 1: commit transaction and disconnect.
```

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 Step 4 through Step 9 for all the limbo transactions. After you commit or abort all of the limbo transactions, they are resolved.
To resolve limbo transactions using PROUTIL:

1. Try to start a database session with PROSERVE, PRO, or PROUTIL. If 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:

   13:27:05 SRV  0: Transaction 760, on coordinator sports 1 #42453, is in a limbo state. (2038)
   13:27:05 SRV  0: The database contains limbo transactions. (2043)
   13:27:05 SRV  0: See list on the log file .lg. Use PROUTIL sports 2 -C 2phase recover. (2042)

   Capture the following information, for all the listed limbo transactions:
   - 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.

2. Enter the following command against the coordinator database to determine if the coordinator committed or aborted the limbo transaction:

   proutil db-name -C 2phase commit tr-number

   Where db-name specifies the coordinator database, and 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.
3. Commit or abort the limbo transactions, depending on whether the coordinator committed or aborted the transaction in Step 2.

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)
```

4. 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 10.0A
  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:

```
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 will receive 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 will 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 \(-\text{maxxids}\) 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 \(-\text{maxxids}\) 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 will result in an error if any JTA transactions exist 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.

- **OpenEdge Replication** — A JTA-enabled database cannot be replicated with OpenEdge Replication.

- **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
```

Your database must be offline when enabling for JTA transactions. Enabling your database for JTA transactions disables after-imaging. You must re-enable after-imaging after enabling the database for JTA transactions. For complete syntax details, see the “`PRUTIL ENABLEJTA QUALIFIER`” section on page 21–52.

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
```

The database must be offline when disabling for JTA transactions.
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 Chapter 20, “PROMON Utility.”

Resolving JTA transactions

Since OpenEdge relinquishes transaction control for JTA transactions, it is dangerous to intervene and manually resolve outstanding JTA transactions. This intervention could compromise the database’s referential integrity 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:

   | MONITOR Release 10.0A                      |
   | 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                      |
   | 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 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:

```
<table>
<thead>
<tr>
<th>Tran Id</th>
<th>Usr</th>
<th>JTA State</th>
<th>XID</th>
</tr>
</thead>
<tbody>
<tr>
<td>1492</td>
<td>5</td>
<td>JTA Prepared</td>
<td>4a982a20-49b7-11da-8cd6-0800200c9a66</td>
</tr>
<tr>
<td>1494</td>
<td>16</td>
<td>JTA Active</td>
<td>0</td>
</tr>
</tbody>
</table>
```

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 Step 3. 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 your database’s referential integrity.

Repeat **Step 4** and **Step 5** for all the outstanding JTA transactions. After you commit or abort all of the transactions, they are resolved.
Part III

Maintaining and Monitoring Your Database

Chapter 14, Managing Performance
Chapter 15, Maintaining Database Structure
Chapter 16, Dumping and Loading
Chapter 17, Logged Data
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. The following sections discuss options for managing database performance:

- Introduction to performance management
- Tools for monitoring performance
- Server performance factors
- Memory usage
- Alternate Buffer Pool
- Operating system resources
- Database fragmentation
- Index use
- 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
- Virtual system tables
- Windows Performance tool

PROMON utility

The OpenEdge Monitor (PROMON) utility helps you monitor database activity and performance. Chapter 20, “PROMON Utility,” 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.

Windows Performance tool

The Windows Performance tool is a graphical tool, supplied with Windows, that lets you monitor the performance of a local or remote Windows workstation. The Performance tool measures the performance of workstation objects like processes and memory using a defined set of counters. The OpenEdge database provides a comprehensive set of counters ranging from measurements about the number of clients running to the number of database records read or written. These counters are derived from the PROMON utility, Summary of Activity option, and they report on the state and performance of a particular database. Database support for the Windows Performance tool does not replace the PROMON utility; it simply provides another mechanism that system administrators can use to monitor performance-related data.

The OpenEdge-specific counters are defined in the registry at installation. Each time you want to monitor database performance, you can specify the type of data that you want to monitor and how you want to monitor it. For instructions on adding OpenEdge-specific counters to the Performance tool log, see the Windows Performance help.
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 the “Disk I/O” section on page 14–5.

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 Chapter 15, “Maintaining Database Structure.”
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. Figure 14–1 illustrates database disk I/O.

![Database I/O Diagram]

Figure 14–1: 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 Step 1 must wait. For this reason, performance is improved if empty buffers are always available. See the “How the database engine writes modified buffers” section on page 14–8 for detailed steps.
Figure 14–2 illustrates how the engine reads a database record into a buffer.

* Asynchronous page writers, if they are running, continuously perform this step in the background.
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 Figure 14–3.

Replacing Unmodified Buffers

Replacing Modified Buffers

1 Eviction (write modified buffer to disk).

2 Place new data in buffer.

Figure 14–3: 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 the “Database buffers” section on page 14–6 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. Figure 14–4 shows the Buffer Hits field in a sample Activity display.

Figure 14–4: 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 Chapter 19, “Database Startup Parameters” 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 Chapter 26, “Virtual System Tables.”
The following ABL code example demonstrates how to turn private read-only buffers on and off:

```ABL
/*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:

```sql
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 Chapter 3, “Starting Up and Shutting Down,” 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 Chapter 21, “PROUTIL Utility.”
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. Figure 14–5 illustrates how an APW scans the BLKTBL chain.

![Figure 14–5: Block Table (BLKTBL) chain](image)

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. Figure 14–6 illustrates the LRU chain.

![LRU Chain Diagram]

**Figure 14–6: 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. **Figure 14–7** 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) |

<table>
<thead>
<tr>
<th></th>
<th>Total</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total DB writes</td>
<td>3</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>APW DB writes</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>scan writes</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>APW queue writes</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>cpq queue writes</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>scan cycles</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>buffers scanned</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>bfs checkpointed</td>
<td>173</td>
<td>0</td>
<td>0.11</td>
<td>0.14</td>
</tr>
<tr>
<td>Checkpoints</td>
<td>82110</td>
<td>0</td>
<td>5.22</td>
<td>6.79</td>
</tr>
<tr>
<td>Marked at checkpoint</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Flushed at checkpoint</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Number of APWs: 1

**Figure 14–7:** PROMON Page Writers Activity display

**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 Chapter 26, “Virtual System Tables.” 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 \texttt{UserTableStat} and \texttt{UserIndexStat} VSTs is established at server startup. By default the first 100 objects are monitored. Table 14–1 describes the startup parameters. You can alter the base during runtime, but not the range.

### Table 14–1: Startup parameters for I/O by object

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Indication</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{-basetable $n$}</td>
<td>Start monitoring tables at table $n$. The default is 1. Change the \texttt{basetable} value during runtime by updating the value of the \texttt{_TableBase} field in the \texttt{StatBase} VST.</td>
</tr>
<tr>
<td>\texttt{-tablerange $n$}</td>
<td>Monitor $n$ tables.(^1) If not specified, the range is 100.</td>
</tr>
<tr>
<td>\texttt{-baseindex $n$}</td>
<td>Start monitoring indexes at index $n$. The default is 1. Change the \texttt{baseindex} value during runtime by updating the value of the \texttt{_IndexBase} field in the \texttt{StatBase} VST.</td>
</tr>
<tr>
<td>\texttt{-indexrange $n$}</td>
<td>Monitor $n$ indexes.(^1) If not specified, the range is 100.</td>
</tr>
</tbody>
</table>

\(^1\) You cannot alter the range after startup.

To consume less memory, set the \texttt{-indexrange} and \texttt{-tablerange} parameters to smaller values.

### 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. Figure 14–8 shows a sample display.

![Figure 14–8: PROMON BI Log Activity display](image)

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

The “Disk I/O” section on page 14–5 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 Chapter 3, “Starting Up and Shutting Down,” 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 the “**Increasing the BI block size**” section on page 14–19.

**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. **Figure 14–9** illustrates the default BI clusters.

![Figure 14–9: BI clusters at startup](image)
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. Figure 14–10 illustrates how BI clusters fill over time.

Figure 14–10: 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 (PRUTIL IDXBUILD)
- Manually (PRUTIL 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 enter the following 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 the “Increasing the BI cluster size” section on page 14–17.

   For detailed information on this command, see Chapter 21, “PRUTIL Utility.”

**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 Chapter 19, “Database Startup Parameters.”
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 Chapter 18, “Startup and Shutdown Commands.”

   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 Chapter 18, “Startup and Shutdown Commands.”

---

**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**. Figure 14–11 shows a sample display.

![Figure 14–11: PROMON AI Log Activity display](image)

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 Chapter 3, “Starting Up and Shutting Down,” 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 the “Providing more BI buffers” section on page 14–17.) 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 Chapter 23, “RFUTIL Utility.”

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 the “Increasing the BI block size” section on page 14–19.

4. Change the AI block size by entering the following command:

   ```
   rfutil db-name -C aimage truncate -aiblocksize size [ -a afilename ]
   ```

   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 Chapter 5, “Backing Up a Database.”

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 Chapter 23, “RFUTIL Utility.”

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
- Controlling memory use
- Shared memory allocation
- Controlling memory use
- Shared memory consumption and allocation impact
- Shared memory summary

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. Figure 14–12 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 14–12: Sample 4GB process address space](image)
Most 32-bit operating systems divide a 4 GB address space into different address ranges or sections. Each section has explicit purposes. In Figure 14–12, 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**

Table 14–2 lists the startup parameters used to fine-tune memory allocation on the server system.

**Table 14–2:  Startup parameters that affect memory allocation**

<table>
<thead>
<tr>
<th>Startup parameter</th>
<th>Suggested use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blocks in Database Buffers (-B)</td>
<td>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.</td>
</tr>
<tr>
<td>Maximum Clients per Server (-Ma)</td>
<td>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.</td>
</tr>
<tr>
<td>Maximum Servers (-Mn)</td>
<td>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 (-M1) parameter.</td>
</tr>
<tr>
<td>Number of Users (-n)</td>
<td>Set this parameter large enough to include both local and remote users.</td>
</tr>
<tr>
<td>Pin Shared Memory (-pinshm)</td>
<td>Use this parameter to prevent the database engine from swapping shared memory contents to disk.</td>
</tr>
</tbody>
</table>

1. Relevant only to database accessed using client/server.

**Shared memory allocation**

In Release 10.1B and forward, shared memory segment allocation methodologies are dynamic. Prior to this release, shared memory segments had a fixed maximum size; now, 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 `-shmsegsize n` server parameter. See the “Shared memory segment size (-shmsegsize)” section on page 19–62 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:

- 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 the “PROMON R&D Advanced Options” section on page 20–24.

### Shared memory consumption and allocation impact

The database data structures placed into shared memory are larger in Release 10.1B and higher than they were in prior releases. In particular, shared pointers are now 64 bits wide for both 64-bit and 32-bit versions of the OpenEdge RDBMS; previously they were 32 bits wide on 32-bit versions. As a result, some data structures, such as database buffer-pool headers and lock table entries, consume noticeably more memory than before.
While the changes in release 10.1B and later have definite advantages, the change in data structure size may cause unexpected errors. Table 14–3 identifies potential problems and presents their solutions.

**Table 14–3: Shared memory problems and solutions**

<table>
<thead>
<tr>
<th>If . . .</th>
<th>And the database log or standard output contains . . .</th>
<th>Then the solution is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>You cannot start the server because data structures do not fit and they did before.</td>
<td>The data space of the process is not enough for the shm segment (1176)</td>
<td>Lower the value of (-B) by 10 percent to overcome this problem if you are close to the maximum amount of shared memory for the system and the value of (-B) is large. Continue reducing (-B) until the database starts successfully.</td>
</tr>
<tr>
<td>The server starts, but clients cannot connect in self-serving mode.</td>
<td>Unable to attach shared memory “segment name” error ##. (1720)</td>
<td>Lower the value of (-shmsegsize).</td>
</tr>
<tr>
<td>This may be caused by lack of sufficient contiguous free address space in combination with the larger default segment size used by the server.</td>
<td></td>
<td>When the address space has several smaller holes, rather one large, the operating system may be able to locate the segments in them. This is likely to happen in Windows, where the system may have mapped .dll files in a manner such that there are no large chunks of free address space.</td>
</tr>
<tr>
<td>A client that connects to several databases in self-serving mode may no longer be able to do so. This can happen due to larger segments.</td>
<td>Unable to attach shared memory “segment name” error ##. (1720)</td>
<td>Lower the value of (-shmsegsize).</td>
</tr>
<tr>
<td>There is a conflict between large shared ABL procedure libraries (that are mapped in a single contiguous memory segment) and large shared memory segments created by the server.</td>
<td></td>
<td>Lower the value of (-shmsegsize).</td>
</tr>
</tbody>
</table>
Shared memory summary

The shared memory addressing enhancements made in OpenEdge Release 10.1B and higher increase the shared memory addressability on 32-bit and 64-bit platforms 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

In Release 10.2B and higher, 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. Details of the Alternate Buffer Pool are discussed in the following sections:

- Overview
- Designating objects to the Alternate Buffer Pool
- Viewing Alternate Buffer Pool selections
- Allocating Alternate Buffer Pool

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
- Specifying objects for the Alternate Buffer Pool

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 CREATE/ALTER 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. A new keyword, BUFFER-POOL is added in Release 10.2B. Because of the new keyword, earlier clients 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
- Viewing Alternate Buffer Pool settings with PROUTIL

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:

```
Quick Alternate Buffer Pool Report

List of object-level assignable objects that are assigned to the alternate buffer pool at the object level
Object       Area
------------------------
Customers    Customer/Order Area
            Customer/Order Area
            Customer Index Area
            Customer Index Area
            Customer Index Area
            Primary Index Area
            Customer/Order Area
Order-Line   Customer/Order Area

List of object-level assignable objects that are assigned to the alternate buffer pool at the area level
Object       Area
------------------------

```

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)`.
Alternate Buffer Pool

The following is an excerpt of VIEWB2 output:

<table>
<thead>
<tr>
<th>Area  8: &quot;Customer/Order Area&quot; - Alternate Buffer Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Enablement</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Alternate</td>
</tr>
<tr>
<td>Default</td>
</tr>
<tr>
<td>Alternate</td>
</tr>
<tr>
<td>Alternate</td>
</tr>
<tr>
<td>Default</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area  9: &quot;Primary Index Area&quot; - Primary Buffer Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Enablement</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Alternate</td>
</tr>
<tr>
<td>Default</td>
</tr>
<tr>
<td>Default</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Area  10: &quot;Customer Index Area&quot; - Primary Buffer Pool</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object Enablement</td>
</tr>
<tr>
<td>--------------------</td>
</tr>
<tr>
<td>Alternate</td>
</tr>
<tr>
<td>Alternate</td>
</tr>
<tr>
<td>Alternate</td>
</tr>
</tbody>
</table>

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 Chapter 19, “Database Startup Parameters” 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 Chapter 21, “PROUTIL Utility” 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.
Figure 14–13 shows how semaphores control access to shared resources.

![Figure 14–13: How a semaphore coordinates concurrent access](image)

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:

\[
#SEM = \text{Max-possible-users} (-n) + \text{Max-possible-servers} (-Mn) + 4
\]
Table 14–4 lists the UNIX kernel parameters that control the number and size of the semaphore sets.

### Table 14–4: UNIX kernel parameters that affect semaphores

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Recommended setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>SEMMNI</td>
<td>The maximum number of semaphore identifiers allowed for the system</td>
<td>One per active multi-user database. If you set this value too low, the database engine might generate error 1131.</td>
</tr>
<tr>
<td>SEMMSL</td>
<td>The maximum number of semaphores allowed per semaphore identifier (semaphore set)</td>
<td>(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.</td>
</tr>
<tr>
<td>SEMMNS</td>
<td>Total number of semaphores allowed for the system</td>
<td>SEMMSL * number of active databases. If you set this value too low, the database engine might generate error 1093, 1131, or 1195.</td>
</tr>
<tr>
<td>SEMMNU</td>
<td>Maximum number of semaphore undo structures allowed for the system</td>
<td>Same value as SEMMNS. If you set this value too low, the database engine might generate error 1081.</td>
</tr>
</tbody>
</table>

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 Chapter 2, “OpenEdge RDBMS Limits” for details on calculating the number of file descriptors required.
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:

1. 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 Chapter 16, “Dumping and Loading.”

2. 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 Chapter 15, “Maintaining Database Structure.”

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-identifier new-limit
```

Table 14–5 describes the PROUTIL qualifiers for changing create and toss limits.

**Table 14–5: PROUTIL qualifiers for changing create and toss limits**

<table>
<thead>
<tr>
<th>Limit</th>
<th>Object type</th>
<th>Qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create</td>
<td>Area</td>
<td>PROUTIL SETAREACREATELIMIT qualifier</td>
</tr>
<tr>
<td></td>
<td>BLOB</td>
<td>PROUTIL SETBLOBCREATELIMIT qualifier</td>
</tr>
<tr>
<td></td>
<td>Table</td>
<td>PROUTIL SETTABLECREATELIMIT qualifier</td>
</tr>
<tr>
<td>Toss</td>
<td>Area</td>
<td>PROUTIL SETAREATOSSLIMIT qualifier</td>
</tr>
<tr>
<td></td>
<td>BLOB</td>
<td>PROUTIL SETBLOBTOSSLIMIT qualifier</td>
</tr>
<tr>
<td></td>
<td>Table</td>
<td>PROUTIL SETTABLETOSSLIMIT qualifier</td>
</tr>
</tbody>
</table>

The current values for create and toss limits are displayed by using PROUTIL DISPTOSSCREATELIMITS. The limits are displayed for a specified area. See the “PROUTIL DISPTOSSCREATELIMITS qualifier” section on page 21–41 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. Table 14–6 describes situations and suggested solutions.

**Table 14–6: Create and toss limit situations and solutions**

<table>
<thead>
<tr>
<th>Situation</th>
<th>Suggested solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fragmentation occurs on updates to existing records. You anticipated one fragment, but two were created.</td>
<td>Increase create limit</td>
</tr>
<tr>
<td>Fragmentation occurs on updates to existing records or you have many blocks on the RM chain with insufficient space to create new records.</td>
<td>Increase toss limit</td>
</tr>
<tr>
<td>There is limited fragmentation, but database block space is being used inefficiently, and records are not expected to grow beyond their original size.</td>
<td>Decrease create limit</td>
</tr>
<tr>
<td>There is limited fragmentation, but database block space is being used inefficiently, and records are not expected to grow beyond their original size.</td>
<td>Decrease toss limit</td>
</tr>
</tbody>
</table>
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
- Compacting indexes
- Rebuilding indexes
- Activating a single index

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. Table 14–7 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 14–7: Factor values

<table>
<thead>
<tr>
<th>Factor range</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 to 2</td>
<td>The index is well-utilized and balanced. You do not have to rebuild it.</td>
</tr>
<tr>
<td>2 to 2.5</td>
<td>The index is less than 50 percent utilized and/or the index is unbalanced. You should consider a rebuild.</td>
</tr>
<tr>
<td>2.5 to 3</td>
<td>The index is less than 25 percent utilized and/or the index is very unbalanced. You should rebuild this index.</td>
</tr>
</tbody>
</table>

### 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 Chapter 21, “PROUTIL Utility.”

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 Chapter 15, “Maintaining Database Structure.”

**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.

### 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)

**Notes:** When you run the Index Rebuild, the database must not be in use.

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 ]
  table [owner-name.]table-name |
  area area-name | schema schema-owner |
  activeindexes | inactiveindexes ]
[ -T dir-name ] | [ -SS sort-file-directory-specification ]
[ -TB blocksize ] | [ -TM n ] | [ -B n ] | [ -SG n ]
[ -threads n ] | [ -threadnum n ]
```

For more information on each option, see the “PROUTIL IDXBUILD qualifier” section on page 21–66.
When you enter this command without the all, table, area, or schema qualifiers, the following menu appears:

<table>
<thead>
<tr>
<th>Index Rebuild Utility</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select one of the following:</td>
</tr>
<tr>
<td>All (a/A) - Rebuild all the indexes</td>
</tr>
<tr>
<td>Some (s/S) - Rebuild only some of the indexes</td>
</tr>
<tr>
<td>By Area (r/R) - Rebuild indexes in selected areas</td>
</tr>
<tr>
<td>By Schema (c/C) - Rebuild indexes by schema owners</td>
</tr>
<tr>
<td>By Table (t/T) - Rebuild indexes in selected tables</td>
</tr>
<tr>
<td>By Activation (v/V) - Rebuild selected active or inactive indexes</td>
</tr>
<tr>
<td>Quit (q/Q) - Quit, do not rebuild</td>
</tr>
</tbody>
</table>

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 Activation** option to select active or inactive indexes.

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:

\[
(size \text{ of one index entry}) \times (number \text{ of records in file}) \times 3
\]
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 used 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 OK. 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.
- Increase the Merge Number (-TM) startup 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.

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:

\[
M = \text{sort-block-size} \times (\text{number-of-simultaneous-sorts+Merge-Number(-TM) parameter})
\]

Index rebuild always requires eight simultaneous sorts, so during index rebuild:

\[
M = \text{sort-block-size} \times (8 + (-\text{TM}) \text{ parameter})
\]

Therefore, in the default case:

\[
M = (2 \times (8+5)) = 26K
\]

**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 PROUNITL IDXACTIVATE. When you run IDXACTIVATE you provide the name of one inactive index on the command line. The general syntax is as follows:

**Syntax**

```bash
proutil db-name -C idxactivate [owner-name.]table-name.index-name
                   [useindex index-name] [recs n] [refresh t]
```

See the “PROUNITL IDXACTIVATE qualifier” section on page 21–64 for the details of the command.

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 proceed. You are given the option of waiting or cancelling the index activation. If you chose to wait, you must wait until all the users with an earlier schema timestamp disconnect, 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. 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
       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
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.
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 Chapter 26, “Virtual System Tables.”
Once you create and start a database, you must manage the database so that it operates efficiently and meets the needs of users. In the following sections, this chapter describes methods to manage the database structure and alter it as necessary to improve storage and performance:

- OpenEdge Structure utility
- OpenEdge Structure Statistics utility
- OpenEdge Structure List utility
- OpenEdge Structure Add utility
- OpenEdge Structure Add Online utility
- Area numbers
- Validating structure description files
- 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 Physical 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
If a full backup has not yet been performed against `service.db`, the message would read “NO FULL BACKUP HAS BEEN DONE.”

---

**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)
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: Control Area, 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 VARIABLE, 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
```
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 the “OpenEdge Structure Add Online utility” section on page 15–11.

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. For example:

   ```
   # 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. 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         start time
   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. 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.
**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: /usr1/cmcguire/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:

   ![User Name and Type Table](image)

   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:

<table>
<thead>
<tr>
<th>Usr Name</th>
<th>Type</th>
<th>Pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>usr1</td>
<td>1234</td>
</tr>
<tr>
<td>102</td>
<td>usr2</td>
<td>5678</td>
</tr>
</tbody>
</table>

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.
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:

   ```
   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 the “Validating structure description files” section on page 15–22.

3. Add the new areas and extents. Use the PROSTRCT Utility with the ADDONLINE qualifier, specifying the `.st` file you created in Step 1. For example:

   ```
   prostrct addonline service add.st
   ```
4. PROSTRCT ADDONLINE adds the new extents or storage areas and extents to the existing database control area, and outputs descriptive information, such as:

<table>
<thead>
<tr>
<th>Formatting extents:</th>
<th>area name</th>
<th>path name</th>
<th>size</th>
<th>path name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary Recovery Area</td>
<td>/user/joe/service.b3 00:00:02</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Primary Recovery Area</td>
<td>/user/joe/service.b4 00:00:02</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Employee</td>
<td>/user/joe/service_7.d5 00:00:02</td>
<td>4096</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Employee</td>
<td>/user/joe/service_7.d6 00:00:04</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Image Area 10</td>
<td>/user/joe/service.a10 00:00:00</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Image Area 11</td>
<td>/user/joe/service.a11 00:00:00</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Image Area 12</td>
<td>/user/joe/service.a12 00:00:00</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Enabling extents:</th>
<th>area name</th>
<th>path name</th>
<th>size</th>
<th>path name</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Primary Recovery Area</td>
<td>/user/joe/service.b3 00:00:00</td>
<td>128</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Primary Recovery Area</td>
<td>/user/joe/service.b4 00:00:00</td>
<td>16</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Employee</td>
<td>/user/joe/service_7.d5 00:00:00</td>
<td>4096</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Employee</td>
<td>/user/joe/service_7.d6 00:00:00</td>
<td>32</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Image Area 10</td>
<td>/user/joe/service.a10 00:00:01</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Image Area 11</td>
<td>/user/joe/service.a11 00:00:00</td>
<td>64</td>
<td></td>
</tr>
<tr>
<td></td>
<td>After Image Area 12</td>
<td>/user/joe/service.a12 00:00:00</td>
<td>16</td>
<td></td>
</tr>
</tbody>
</table>

5. 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
- Trimming unused area memory
- Error checking

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.
Figure 15–1 shows the resulting area array where the areas are numbered sequentially.

![Sequential area allocation diagram]

**Figure 15–1:** 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.
Figure 15–2 shows the allocation of the area array for the database with areas numbered 1000, 2000, and 3000.

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 the “Trimming unused area memory” section on page 15–19 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 the “Maximum area number (-maxAreas)” section on page 19–46.

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:

```plaintext
# 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 .
#```
Figure 15–3 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.

![Area array layout](image)

**Figure 15–3: 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

- Adding areas online

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 the “Creating a structure description file” section on page 1–3.
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
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.

To remove storage areas:

1. 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:

   ```bash
   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:

   ```bash
   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 the “PROUTIL TRUNCATE AREA qualifier” section on page 21–103.

2. Use PROSTRCT REMOVE to remove extents from the storage area. For example:

   ```bash
   prostrct remove db-name extent-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:

   ```bash
   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-name table-area
[ index-area ]
```

**Notes:** For the complete syntax description, see Chapter 21, “PROUTIL Utility.”

The _UserStatus virtual system table displays the utility’s progress. For more information see Chapter 26, “Virtual System Tables.”

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 Chapter 21, “PROUTIL Utility.”
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 Chapter 21, “PROUTIL Utility.”

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 Chapter 21, “PROUTIL Utility”.
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 Chapter 26, “Virtual System Tables.”

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:

```abl
/* 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:

```sql
SELECT * FROM _UserStatus WHERE _UserStatus-Operation = "Index move";
```
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, in the following sections:

- Overview of dumping and loading
- Dumping ABL database definitions
- Dumping database contents
- Loading database definitions
- Loading database contents
- Bulk loading
- Reconstructing bad load records
- Specialized dump and load techniques

Note: For a complete description of dumping and loading SQL content, see the SQLDUMP and SQLLOAD descriptions in Chapter 25, “SQL Utilities.”
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 Chapter 8, “Maintaining Security,” 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.

Table 16–1 shows the definitions that are dumped in each case.

**Table 16–1: Definitions dumped to the definition file**

<table>
<thead>
<tr>
<th>Definitions</th>
<th>All tables</th>
<th>Selected tables</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables, fields, and indexes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Sequence definitions</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Auto-connect records</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Collation</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>_User</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

If you dump individual tables, you must also dump the sequence definitions and auto-connect records separately. See the “Dumping sequence definitions” section on page 16–7 and the “Dumping auto-connect records” section on page 16–8 for instructions.

**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. Figure 16–1 shows a sample data definitions file for a single table.

```
ADD TABLE "Customer"
  AREA "Info Area"
  DESCRIPTION "Customer information"
  DUMP-NAMES "customer"
  TABLE-TRIGGER "CREATE" NO-OVERRIDE PROCEDURE "sports/crecust.p" CRC "?"
  TABLE-TRIGGER "DELETE" NO-OVERRIDE PROCEDURE "sports/delcust.p" CRC "?"
  TABLE-TRIGGER "UPDATE" NO-OVERRIDE PROCEDURE "sports/updcust.p" CRC "?"

ADD FIELD "Cust-Num" OF "Customer" AS integer
  FORMAT "<<<0"
  INITIAL "0"
  POSITION 1
  MAX-WIDTH 4
  VALUE "cust-num > 0"
  VALUE "Customer number must be greater than zero"
  ORDER 10

ADD FIELD "Name" OF "Customer" AS character
  FORMAT "X[20]"
  INITIAL ""
  POSITION 1
  MAX-WIDTH 40
  ORDER 90

ADD FIELD "Address" OF "Customer" AS character
  FORMAT "X[40]"
  INITIAL ""
  POSITION 4
  MAX-WIDTH 40
  ORDER 40

ADD INDEX "Cust-Num" ON "Customer"
  AREA "Info Area"
  UNIQUE
  PRIMARY
  INDEX-FILE "Cust-Num" ASCENDING

ADD INDEX "Comments" ON "Customer"
  AREA "Info Area"
  WORD INDEX-FILE "Comments" ASCENDING

PSC
  cputem=1300659-1
  00000000-88
```

Figure 16–1:  Sample data definitions file
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. Table 16–2 explains these values.

### Table 16–2: Data definitions file trailer values

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>codepage=codepage</td>
<td>The code page used by the database. For more information, see <em>OpenEdge Development: Internationalizing Applications</em>.</td>
</tr>
<tr>
<td>encpolicy=yes</td>
<td>Indicates that there are encryption policy settings in the data definition file.</td>
</tr>
<tr>
<td>bufpool=yes</td>
<td>Indicates that there are Alternate Buffer Pool settings in the data definition file.</td>
</tr>
<tr>
<td>Character count</td>
<td>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.</td>
</tr>
</tbody>
</table>

### 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.

**Notes:** 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 and Loading

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 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 Chapter 25, “SQL Utilities.”

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 the “PROUTIL IDXANALYS qualifier” section on page 21–65.

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 the “PROUTIL DUMP qualifier” section on page 21–42.
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.bdn. 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:

<table>
<thead>
<tr>
<th>Header</th>
<th>Record length</th>
<th>Table number</th>
<th>Binary record</th>
<th>Record CRC</th>
</tr>
</thead>
</table>

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:

**Syntax**

```bash
proutil db-name -C dumpspecified [owner-name.]table-name.field-name
   operator1 low-value AND operator2 high-value directory [-preferidx index-name]
```

Table 16–3 describes selective dump syntax.

<table>
<thead>
<tr>
<th>Table 16–3: DUMPSPECIFIED syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Field</td>
</tr>
<tr>
<td><code>db-name</code></td>
</tr>
<tr>
<td><code>owner-name</code></td>
</tr>
<tr>
<td><code>table-name</code></td>
</tr>
<tr>
<td><code>field-name</code></td>
</tr>
</tbody>
</table>
Note: At Release 10.1C, the dumpspecified qualifier permits you to selectively dump records in a specified range. Using this qualifier, you can dump data from a specific table by specifying a record that falls between two values. For the complete syntax for selective binary dumps, see the “PROUTIL DUMPSPESIFIED qualifier” section on page 21–45.

**Examples**

The following syntax dumps all order date values greater than 2/3/02 from the Sports2000 database:

**Syntax**

```
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:

**Syntax**

```
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:

**Syntax**

```bash
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:

**Syntax**

```bash
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 the “Dumping sequence values with a Data tool” section on page 16–16 for more information.

### Contents file format

Figure 16–2 shows a sample contents file.

```
1 "USA" "Lift Line Skiing " "276 North Street " "Boston " "MA" " 02114 " "Gloria Shepley " "(617) 450-0087" "HXM" "66700 42568 "Net30" 35 "This customer is on credit hold . Last payment received marked "Insufficient Funds "!"
.
.
84 "USA" "Spike 's Volleyball " "34 Dudley St " "Genoa " "NV" "89411" "Craig Eleazer " "(702) 272-9264" "KIK" "20400 18267 "Net30" 5 ""
.
.
```

- Indicates end of contents
- Always specified

```
PSC
filename =Customer
records =00000083
ldbnme =junk
timestamp =2000/06/28-16:20:51
numformat =thousands-, fractional -separator
dateformat =mdy-1900
map=NO-MAP
codepage =ibm850
.
```

- Indicates end of variables
- Character count (always the last line)

**Figure 16–2:** 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.
Table 16–4 explains these variables. If the database uses the default values for these variables, these variables do not appear in the contents file.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename=table-name</td>
<td>Table name.</td>
</tr>
<tr>
<td>records=num-records</td>
<td>Number of records contained in the file.</td>
</tr>
<tr>
<td>ldbname=logical-db-name</td>
<td>Logical database name of the source database.</td>
</tr>
<tr>
<td>time stamp=time stamp</td>
<td>Time stamp of the database.</td>
</tr>
<tr>
<td>num format=thousands-separator, fractional-separator</td>
<td>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.</td>
</tr>
<tr>
<td>date format=date format</td>
<td>The date format used by the database contents.</td>
</tr>
<tr>
<td>map=YES protermcap-entry</td>
<td>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.</td>
</tr>
<tr>
<td>codepage=codepage</td>
<td>The code page used by the database. For more information, see OpenEdge Development: Internationalizing Applications.</td>
</tr>
</tbody>
</table>

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.

1. Information used when loading the file. If the value is specified in the trailer and is specified incorrectly, the data will not load.
Dumping and Loading

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 Chapter 25, “SQL Utilities.”
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 Chapter 25, “SQL Utilities.”

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:

**Syntax**

```
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 choose to build indexes on your data as it loads with the `build indexes` parameter. For complete syntax details, see the “PROUTIL LOAD qualifier” section on page 21–85.

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 the “Loading sequence values using a **Data tool**” section on page 16–24 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 Chapter 25, “SQL Utilities.”
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 the “Dumping user table contents with a Data tool” section on page 16–16.

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.
Figure 16–3 shows a description file created with the Data Administration tool.

![Diagram of Table name, Data (.d) file, and Error file]

**Table name**
- customer
- Name
- Address
- City
- Zip
- Phone
- Contact
- Sales-rep
- Sales-region
- Max-credit
- Curr-bal
- Terms
- Tax-no
- Discount
- Mnth-sales
- Ytd-sls

**Field names**

**Figure 16–3: 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. Table 16–5 lists the tasks and actions.

**Table 16–5: Modifying the description file**

<table>
<thead>
<tr>
<th>Task</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Include comments in the .fd file.</td>
<td>Place a pound sign (#) at the beginning of each line.</td>
</tr>
<tr>
<td>Include the description of multiple tables in one description file.</td>
<td>Separate tables with a period on a line by itself.</td>
</tr>
<tr>
<td>Skip a field.</td>
<td>Replace the field name with a caret (^).</td>
</tr>
</tbody>
</table>

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. Figure 16–4 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
```

Figure 16–4: Example modified Bulk Loader description file

**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 the “Creating a Bulk Loader description file” section on page 16–25.

2. Verify that the table definitions are in the database.

3. Run the Bulk Loader utility:

   ```bash
   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 the “PROUTIL BULKLOAD qualifier” section on page 21–18 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.

Figure 16–5 illustrates how the OpenEdge Database Utility (PRODB) copies the starting version of a database.

![Figure 16–5: Copying the starting version of a database](image)

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.

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 Chapter 21, “PROUTIL Utility.”

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 Chapter 21, “PROUTIL Utility,” 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:

```plaintext
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.
Specialized dump and load techniques

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 ENKEY 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 Chapter 19, “Database Startup Parameters.”

**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. See Chapter 21, “PROUTIL Utility,” for more information about activating indexes with PROUTIL IDXBUILD.

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.
To deactivate an individual index or all of the indexes using a Data Tool:

1. Access the Data tool.
2. Choose Utilities→Deactivate Indexes. The Index Deactivation dialog box appears.
3. Choose OK. The Data tool lists the tables in the database.
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.

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.

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."
```
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, as described in the following sections:

- OpenEdge Release 10 database log file
- Managing log file size
- Saving key database events
- Event logging in Windows
- Client database-request statement caching
OpenEdge Release 10 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
```

Table 17–1 describes the fields of each entry in the log file. All fields, with the exception of the message text, have a fixed width.

<table>
<thead>
<tr>
<th>Field in log entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>yy/mm/dd</td>
<td>Date in year/month/day format.</td>
</tr>
<tr>
<td>hh:mm:ss.uuu</td>
<td>Local time in hours, minutes, seconds, milliseconds.</td>
</tr>
<tr>
<td>sshhmm</td>
<td>Time zone relative to GMT where s is a + or - and hhm are hours and minutes. For example, a value of -0500 represents a time zone five hours behind GMT.</td>
</tr>
<tr>
<td>P-nnnnnn</td>
<td>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.</td>
</tr>
<tr>
<td>T-nnnnn</td>
<td>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. Space is allocated for a maximum of 5 digits in the thread id.</td>
</tr>
</tbody>
</table>
| S                   | The severity level of the message. There are three possible values:  
  • I — Informational message  
  • W — Warning message  
  • F — Fatal message |
<table>
<thead>
<tr>
<th>Field in log entry</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name nnn:</td>
<td>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.</td>
</tr>
<tr>
<td>(nnnnn)</td>
<td>The message number of the message. More details about the message can be obtained by accessing Help→Messages... from the OpenEdge Desktop.</td>
</tr>
<tr>
<td>Message text...</td>
<td>The text of the message.</td>
</tr>
</tbody>
</table>
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.”
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
- Saving key events
- Enabling save key events
- Disabling save key events
- Stored key events
- `_KeyEvt` table

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:

<table>
<thead>
<tr>
<th>User Control:</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
</tr>
<tr>
<td>0 docdemo</td>
</tr>
<tr>
<td>5 docdemo</td>
</tr>
<tr>
<td>6 KERD</td>
</tr>
</tbody>
</table>

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” section on page 21–53 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 Chapter 22, “PROSTRCT Utility.”
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 the “PRUTIL DISABLEKEYEVENTS” section on page 21–40 for details on the utility.

Stored key events

Table 17–2 lists key database events and the records stored in your database when your database is enabled to save key events.

### Table 17–2: Key events and stored records

<table>
<thead>
<tr>
<th>Database activity</th>
<th>Key events record stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database startup</td>
<td>• Date and time of startup</td>
</tr>
<tr>
<td></td>
<td>• Database startup parameters, one parameter per key event</td>
</tr>
<tr>
<td></td>
<td>record</td>
</tr>
<tr>
<td></td>
<td>• Current database high water mark</td>
</tr>
<tr>
<td>Database shutdown</td>
<td>Date and time of shutdown</td>
</tr>
<tr>
<td>Lock table overflow</td>
<td>Date and time of overflow</td>
</tr>
<tr>
<td>Protrace generation</td>
<td>• Date and time the protrace is generated</td>
</tr>
<tr>
<td></td>
<td>• Name and location of the protrace file</td>
</tr>
<tr>
<td>DBTOOL</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Data and time the utility ended</td>
</tr>
<tr>
<td>PROBKUP</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Location of backup file</td>
</tr>
<tr>
<td></td>
<td>• Size of backup file</td>
</tr>
<tr>
<td></td>
<td>• Data and time the utility ended</td>
</tr>
<tr>
<td>PROCOPY</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Source database location</td>
</tr>
<tr>
<td></td>
<td>• Target database location</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
<tr>
<td>PROLOG</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROMON RESOLVE LIMBO TRANSACTION</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Transaction id</td>
</tr>
<tr>
<td></td>
<td>• Action (commit, abort)</td>
</tr>
</tbody>
</table>
**Table 17–2: Key events and stored records**

<table>
<thead>
<tr>
<th>Database activity</th>
<th>Key events record stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROREST</td>
<td>• Date and time the utility started&lt;br&gt;• Location of backup file&lt;br&gt;• Data and time the utility ended</td>
</tr>
<tr>
<td>PROSTRCT ADD</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROSTRCT BUILDB</td>
<td>• Date and time the utility started&lt;br&gt;• Name of the structure description (.st) file</td>
</tr>
<tr>
<td>PROSTRCT CREATE</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROSTRCT REMOVE</td>
<td>• Date and time the utility started&lt;br&gt;• Names and locations of the deleted extents</td>
</tr>
<tr>
<td>PROSTRCT REORDER</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td>PROSTRCT REPAIR</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROSTRCT UNLOCK</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL 2PHASE BEGIN</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL 2PHASE COMMIT</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL 2PHASE END</td>
<td>Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL 2PHASE MODIFY</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL 2PHASE RECOVER</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL BIGROW</td>
<td>• Date and time the utility started&lt;br&gt;• Number of clusters&lt;br&gt;• Options specified (-r)&lt;br&gt;• Data and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL BULKLOAD</td>
<td>• Date and time the utility started&lt;br&gt;• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL CONV910</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL CONVCHAR (convert option only)</td>
<td>• Date and time the utility started&lt;br&gt;• Codepage specified</td>
</tr>
<tr>
<td>PROUTIL CONVFILE (convert using option only)</td>
<td>• Date and time the utility started&lt;br&gt;• Name and location of the file being converted</td>
</tr>
<tr>
<td>PROUTIL DBAUTHKEY</td>
<td>Date and time the utility started</td>
</tr>
</tbody>
</table>
### Table 17–2: Key events and stored records

<table>
<thead>
<tr>
<th>Database activity</th>
<th>Key events record stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROUTIL DISABLEAUDITING</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL DISABLEKEYEVENTS</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL DISABLEJTA</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL DUMP</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Name of table being dumped</td>
</tr>
<tr>
<td></td>
<td>• Target directory of the dump</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL DUMPSPECIFIED</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Name of table being dumped</td>
</tr>
<tr>
<td></td>
<td>• Name of index field used for selecting records</td>
</tr>
<tr>
<td></td>
<td>• Target directory of the dump</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL ENABLEAUDITING</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL ENABLEKEYEVENTS</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL ENABLELARGEFILES</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL ENABLEJTA</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL ENABLEPDR</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL IDXACTIVATE</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL IDXBUILD</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Options specified (–TB, –TM, –SG)</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL IDXCHECK</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>PROUTIL IDXCOMPACT</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Table name and index name of the index being compacted</td>
</tr>
<tr>
<td></td>
<td>• Degree of compaction</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL IDXFIX</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
</tbody>
</table>
### Table 17–2: Key events and stored records

<table>
<thead>
<tr>
<th>Database activity</th>
<th>Key events record stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>PROUTIL IDXMOVE</td>
<td>• Date and time the utility started&lt;br&gt;• Table and index name of the index being moved&lt;br&gt;• Target area name where the index is being moved&lt;br&gt;• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL LOAD</td>
<td>• Date and time the utility started&lt;br&gt;• Name of file being loaded&lt;br&gt;• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL MVSCH</td>
<td>Date and time the utility started.</td>
</tr>
<tr>
<td>PROUTIL TABLEMOVE</td>
<td>• Date and time the utility started&lt;br&gt;• Name of the table being moved&lt;br&gt;• Target area name where the table is being moved&lt;br&gt;• Target area where indexes with on the table are optionally moved&lt;br&gt;• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL TRUNCATE AREA</td>
<td>• Date and time the utility started&lt;br&gt;• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL TRUNCATE BI</td>
<td>• Date and time the utility started&lt;br&gt;• Option specified (-bi)&lt;br&gt;• Date and time the utility ended</td>
</tr>
<tr>
<td>PROUTIL TRUNCATE BI -F</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>(Force access to damaged database)</td>
<td></td>
</tr>
<tr>
<td>PROUTIL UPDATEVST</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>RFUTIL AIMAGE BEGIN</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>RFUTIL AIMAGE END</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>RFUTIL AIMAGE AIOFF</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>RFUTIL AIMAGE EXTENT EMPTY</td>
<td>• Date and time the utility started&lt;br&gt;•Extent number or pathname of extent to empty</td>
</tr>
<tr>
<td>RFUTIL AIMAGE EXTRACT</td>
<td>• Date and time the utility started&lt;br&gt;• Name of the after-image extent having its notes extracted&lt;br&gt;• Date and time the utility ended</td>
</tr>
<tr>
<td>RFUTIL AIMAGE NEW</td>
<td>Date and time the utility started</td>
</tr>
</tbody>
</table>
Table 17–2: Key events and stored records

<table>
<thead>
<tr>
<th>Database activity</th>
<th>Key events record stored</th>
</tr>
</thead>
<tbody>
<tr>
<td>RFUTIL AIMAGE TRUNCATE</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Specified block size (-aiblocksiz)</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
<tr>
<td>RFUTIL MARK BACKEDUP</td>
<td>Date and time the utility started</td>
</tr>
<tr>
<td>RFUTIL ROLL FORWARD</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Name of the after-image extent to roll forward</td>
</tr>
<tr>
<td></td>
<td>• Specified option (-r)</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
<tr>
<td>RFUTIL ROLL FORWARD RETRY</td>
<td>• Date and time the utility started</td>
</tr>
<tr>
<td></td>
<td>• Name of the after-image extent to roll forward</td>
</tr>
<tr>
<td></td>
<td>• Specified option (-r)</td>
</tr>
<tr>
<td></td>
<td>• Date and time the utility ended</td>
</tr>
</tbody>
</table>

_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
- _KeyEvt_ schema
- _KeyEvt_ table maintenance

_**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. Table 17–3 describes the active fields of the _KeyEvt table.

Table 17–3: _KeyEvt table schema

<table>
<thead>
<tr>
<th>Field name</th>
<th>Description</th>
<th>Data type</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>_KeyEvt-Date</td>
<td>The date and time, in DATETIME-TZ format, when the event occurred</td>
<td>DATETIME-TZ</td>
<td>99/99/9999 HH:MM:SS.SSS+HH:M</td>
</tr>
<tr>
<td>_KeyEvt-Procid</td>
<td>The process id of the process that logged the key event message</td>
<td>INTEGER</td>
<td>-&gt;,&gt;&gt;&gt;,&gt;&gt;&gt;9</td>
</tr>
<tr>
<td>_KeyEvt-Threadid</td>
<td>The thread of the process that logged the key event message</td>
<td>INTEGER</td>
<td>-&gt;,&gt;&gt;&gt;,&gt;&gt;&gt;9</td>
</tr>
<tr>
<td>_KeyEvt-MsgId</td>
<td>The message number of the event</td>
<td>INTEGER</td>
<td>-&gt;,&gt;&gt;&gt;,&gt;&gt;&gt;9</td>
</tr>
<tr>
<td>_KeyEvt-Usrtype</td>
<td>The type of user that logged the event, for example, BROKER, DBUTIL, KEVTRDR, Usr, etc.</td>
<td>CHARACTER</td>
<td>X(32)</td>
</tr>
<tr>
<td>_KeyEvt-Usrnum</td>
<td>The user number of the user that logged the event</td>
<td>INTEGER</td>
<td>-&gt;,&gt;&gt;&gt;,&gt;&gt;&gt;9</td>
</tr>
<tr>
<td>_KeyEvt-Event</td>
<td>The message text associated with the key event</td>
<td>CHARACTER</td>
<td>X(200)</td>
</tr>
</tbody>
</table>

_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 Table 17–4.

### Table 17–4: Event logging components

<table>
<thead>
<tr>
<th>This component . . .</th>
<th>Performs this function . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Event Viewer</td>
<td>Enables users to view the Event Log.</td>
</tr>
<tr>
<td>The Event Log</td>
<td>Records event information.</td>
</tr>
<tr>
<td>PROMSGS.DLL</td>
<td>Contains database messages for the event log.</td>
</tr>
<tr>
<td>CATEGORY.DLL</td>
<td>Contains the 14 categories into which database messages might fall.</td>
</tr>
<tr>
<td>PROMSGS file</td>
<td>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.</td>
</tr>
</tbody>
</table>

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**).

Table 17–5 lists valid event level values.

### Table 17–5: Event level values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>No events are written to the event log.</td>
</tr>
<tr>
<td>Brief</td>
<td>Messages defined as Error and Warning messages are written to the event log.</td>
</tr>
<tr>
<td>Normal</td>
<td>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.</td>
</tr>
<tr>
<td>Full</td>
<td>Every message generated is written to the Event Log. Any messages generated using the Message Statement are also written to the log file.</td>
</tr>
</tbody>
</table>
Understanding the event log components

The components of the Event Log are standards defined by Windows. Table 17–6 describes the Event Log columns.

**Table 17–6: Event Log components**

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Date the event occurred.</td>
</tr>
<tr>
<td>Time</td>
<td>Time the event occurred.</td>
</tr>
<tr>
<td>Source</td>
<td>Source of the event. If the source of the event is a connected OpenEdge database, the database’s name appears in this column.</td>
</tr>
<tr>
<td>Category</td>
<td>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).</td>
</tr>
<tr>
<td>Event</td>
<td>The OpenEdge message number that was generated. These are the same message numbers that are displayed in the standard database .lg file.</td>
</tr>
<tr>
<td>User</td>
<td>Identifies the user logged in to the Windows workstation where the event occurred.</td>
</tr>
<tr>
<td>Computer</td>
<td>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.</td>
</tr>
</tbody>
</table>

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 .dlls 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 .dlls 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 tracks client activity and identifies line numbers within an ABL program (or SQL statement) associated with database requests.

Note: For more information on parameters associated with client database-request statement caching, see Chapter 20, “PROMON Utility”.

The database-request statement cache is updated by individual clients, and is refreshed each time a client performs a database operation. This cache displays ABL and SQL activity, including:

- ABL program names and line numbers of ABL code executing database requests
- ABL program stacks of up to 32 program names and line numbers, beginning with the current program name and line number executing a database operation
- Recent SQL statements

Client database-request statement caching for SQL provides only SQL statements that generate database server requests; for ABL, client database-request statement caching provides program names and line numbers of ABL code that generated database server requests.

Multi-database queries are often complicated by large amounts of data and the physical distribution of such information. Querying multiple data sources typically produces complete result sets. Processing large amounts of data results in poor query response time, especially when considering the physical distribution of the data. To expedite the distribution of these data structures, client database-request statement caching can be used to explicitly monitor ABL and SQL database activity and identify areas within an ABL program (or SQL statements) associated with database requests.

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`. 
Part IV

Reference

Chapter 18, Startup and Shutdown Commands
Chapter 19, Database Startup Parameters
Chapter 20, PROMON Utility
Chapter 22, PROSTRCT Utility
Chapter 21, PROUTIL Utility
Chapter 23, RFUTIL Utility
Chapter 24, Other Database Administration Utilities
Chapter 25, SQL Utilities
Chapter 26, Virtual System Tables
This chapter describes the OpenEdge RDBMS startup and shutdown commands in alphabetical order. It describes the purpose, syntax, and primary parameters for each command, as described in the following sections:

- Startup command syntax
- Database startup and shutdown commands
- PROAIW command
- PROAPW command
- PROBIW command
- PROQUIET command
- PROSERVE command
- PROSHUT command
- PROWDOG command

For a complete list and description of all parameters you can specify with these commands, see Chapter 19, “Database Startup Parameters.”
The conventions used in command syntax descriptions is as follows:

\[ \text{command} \ [ \text{db-name} ] \ [ \text{parameter} ] \ [ \text{value} ] \ldots \]

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
```

Table 18–1 describes each of the command components.

<table>
<thead>
<tr>
<th>Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>command</td>
<td>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.</td>
</tr>
<tr>
<td>db-name</td>
<td>Name of the database you want to connect to.</td>
</tr>
<tr>
<td>parameter, qualifier</td>
<td>Operating criteria for the command.</td>
</tr>
<tr>
<td>value</td>
<td>Numeric value or file specification for the parameter.</td>
</tr>
</tbody>
</table>

**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. Table 18–2 summarizes the tasks performed by each command.

Table 18–2: Database startup and shutdown commands

<table>
<thead>
<tr>
<th>To . . .</th>
<th>Use . . .</th>
</tr>
</thead>
</table>
| 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) | proapw db-name |
| Start a before-image writer (BIW) | probiw db-name |
| Start an after-image writer (AIW) | proaiw db-name |
| Stop all writes to database files by enabling a “quiet” processing point | 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. |

1. Option available only on Enterprise Database products.
# 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 Chapter 14, "Managing Performance.”

## 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 Chapter 14, "Managing Performance."

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 the Chapter 20, "PROMON utility."

- 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 Chapter 14, “Managing Performance.”

**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 | disable } | 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.

`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 Chapter 5, “Backing Up a Database,” and Chapter 14, “Managing Performance.”

**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 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.
- Quiet points are only available on Enterprise systems.
**Examples**

- 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 Explorer or Progress Explorer configuration tools, which saves them in the `conmgr.properties` file.

`parameters`

Specifies the startup parameters for the broker/server. See Chapter 19, “Database Startup Parameters,” 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 servers.

- You create server groups using OpenEdge Explorer or Progress Explorer configuration tools, which save them 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 Explorer or Progress Explorer. For more information on OpenEdge Explorer or Progress 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 username`

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. Table 18–3 describes the shutdown timeout parameter.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>immed</td>
<td>Shutdown occurs in approximately 60 seconds, regardless of transaction backout status</td>
<td>–</td>
</tr>
<tr>
<td>maximum</td>
<td>Shutdown occurs in a maximum of 24 hours, after all no</td>
<td>–</td>
</tr>
<tr>
<td>n</td>
<td>Specifies the shutdown timeout value in seconds</td>
<td>60 - 86400</td>
</tr>
<tr>
<td>ns</td>
<td>Specifies the shutdown timeout value in seconds</td>
<td>60 - 86400</td>
</tr>
<tr>
<td>nm</td>
<td>Specifies the shutdown timeout value in minutes</td>
<td>1 - 1440</td>
</tr>
<tr>
<td>nh</td>
<td>Specifies the shutdown timeout value in hours</td>
<td>1 - 24</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Prompts you for the number of the user you want to disconnect.</td>
</tr>
<tr>
<td>2</td>
<td>Disconnects all users and shuts down the database.</td>
</tr>
</tbody>
</table>
| 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.

```
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.
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.

Specifically, this chapter contains the following sections:

- Issuing startup parameters
- Database server performance parameters
- Database server-type parameters
- Database server internationalization parameters
- Database server statistics parameters
- Database server network parameters
- Database server SSL connection 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.

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:

**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 Table 19–1 to optimize server performance.

Table 19–1: Server performance parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>After-image Buffers</td>
<td>-aibufs</td>
<td>Specifies the number of after-image buffers when running AIW</td>
</tr>
<tr>
<td>After-image Stall</td>
<td>-aistall</td>
<td>Suspends database activity when an empty after-image (AI) file is unavailable</td>
</tr>
<tr>
<td>Blocks in Database Buffers</td>
<td>-B n</td>
<td>Specifies the number of blocks in the database buffers</td>
</tr>
<tr>
<td>Blocks in Database Alternate Buffer Pool</td>
<td>-B2 n</td>
<td>Specifies the number of blocks in the database Alternate Buffer Pool.</td>
</tr>
<tr>
<td>Before-image Buffers</td>
<td>-bibufs</td>
<td>Specifies the number of before-image buffers when running BIW</td>
</tr>
<tr>
<td>Threshold Stall</td>
<td>-bistall</td>
<td>Quiets a database and sends a message to the log file when the recovery log threshold is reached; use with -bithold</td>
</tr>
<tr>
<td>Recovery Log Threshold</td>
<td>-bithold</td>
<td>Specifies the maximum size of the recovery log files in MB</td>
</tr>
<tr>
<td>Direct I/O</td>
<td>-directio</td>
<td>Opens all files in unbuffered mode</td>
</tr>
<tr>
<td>Event Level</td>
<td>-evtlevel</td>
<td>Specifies the level of information written to the Windows Application Event Log</td>
</tr>
<tr>
<td>Before-image Cluster Age</td>
<td>-G n</td>
<td>Specifies the number of seconds before the database reuses a before-image cluster</td>
</tr>
<tr>
<td>Group Delay</td>
<td>-groupdelay</td>
<td>Specifies the number of milliseconds a transaction waits before committing</td>
</tr>
<tr>
<td>Hash Table Entries</td>
<td>-hash</td>
<td>Specifies the number of hash table entries for the buffer pool</td>
</tr>
<tr>
<td>No Crash Protection</td>
<td>-i</td>
<td>Runs the database without integrity or recovery</td>
</tr>
<tr>
<td>Increase parameters online</td>
<td>-increaseto</td>
<td>Increases certain predefined server startup parameters while the database is online</td>
</tr>
<tr>
<td>Lock Table Entries</td>
<td>-L n</td>
<td>Specifies the number of entries in the record locking table</td>
</tr>
<tr>
<td>Lock release</td>
<td>-lkrela</td>
<td>Use the original lock release mechanism installed from previous OpenEdge versions</td>
</tr>
<tr>
<td>Maximum area number</td>
<td>-maxAreas</td>
<td>Specify the maximum area number available</td>
</tr>
</tbody>
</table>
### Table 19–1: Server performance parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Delayed BI File Write</td>
<td>-Mf $n$</td>
<td>Delays writing the last before-image (BI) file records</td>
</tr>
<tr>
<td>VLM Page Table Entry Optimization(^1)</td>
<td>-Mpte</td>
<td>Allocates shared memory in multiples of 8MB for VLM64 support</td>
</tr>
<tr>
<td>Shared-memory Overflow Size</td>
<td>-Mxs $n$</td>
<td>Replaces the default value of the shared-memory overflow area</td>
</tr>
<tr>
<td>Number of Users</td>
<td>-n $n$</td>
<td>Specifies the maximum number of users connected to the database</td>
</tr>
<tr>
<td>Pin Shared Memory</td>
<td>-pinshm</td>
<td>Prevents the OS from swapping shared memory contents to disk</td>
</tr>
<tr>
<td>Semaphore Sets(^2)</td>
<td>-semsets $n$</td>
<td>Changes the number of semaphore sets available to a broker</td>
</tr>
<tr>
<td>Shared memory segment size</td>
<td>-shmsegsize $n$</td>
<td>Specify the maximum shared memory segment size</td>
</tr>
<tr>
<td>Spin Lock Retries</td>
<td>-spin $n$</td>
<td>Specifies the number of times a process tries to acquire a latch before pausing</td>
</tr>
</tbody>
</table>

1. Compaq Tru64 only.
2. UNIX only.
Database server-type parameters

Use the parameters listed in Table 19–2 to start a particular type of server.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auto Server</td>
<td>-m1</td>
<td>Starts an auto server. Used internally by the Database broker</td>
</tr>
<tr>
<td>Manual Server</td>
<td>-m2</td>
<td>Manually starts a server after you start a broker</td>
</tr>
<tr>
<td>Secondary Login Broker</td>
<td>-m3</td>
<td>Starts a secondary broker</td>
</tr>
</tbody>
</table>
Database server internationalization parameters

Use the parameters listed in Table 19–3 to control the format in which data appears.

**Table 19–3:  Server internationalization parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conversion Map</td>
<td>-convmap filename</td>
<td>Identifies the conversion map file</td>
</tr>
<tr>
<td>Case Table</td>
<td>-cpcase tablename</td>
<td>Identifies the case table that establishes case rules for the code page</td>
</tr>
<tr>
<td>Collation Table</td>
<td>-cpcoll tablename</td>
<td>Identifies a collation table to use with the code page</td>
</tr>
<tr>
<td>Internal Code Page</td>
<td>-cpinternal codepage</td>
<td>Identifies the code page used in memory</td>
</tr>
<tr>
<td>Log File Code Page</td>
<td>-cplog codepage</td>
<td>Identifies the code page used for writing messages to the log file</td>
</tr>
<tr>
<td>Print Code Page</td>
<td>-cpprint codepage</td>
<td>Identifies the code page used for printer output</td>
</tr>
<tr>
<td>R-code in Code Page</td>
<td>-cprcodein codepage</td>
<td>Identifies the code page used for reading r-code text segments</td>
</tr>
<tr>
<td>Stream Code Page</td>
<td>-cpstream codepage</td>
<td>Identifies the code page used for stream I/O</td>
</tr>
<tr>
<td>Terminal Code Page</td>
<td>-cpterm codepage</td>
<td>Identifies the code page for character terminals</td>
</tr>
</tbody>
</table>
Database server statistics parameters

Use the parameters listed in Table 19–4 to collect statistics for table and index access.

Table 19–4: Server statistics collection parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Base Index</td>
<td>-baseindex n</td>
<td>Specifies a range of indexes for which you want to collect statistics; use with -indexrangesize</td>
</tr>
<tr>
<td>Base Table</td>
<td>-basetable n</td>
<td>Specifies a starting table number in a range of tables for which you want to track access statistics; use with -tablerangesize</td>
</tr>
<tr>
<td>Index Range Size</td>
<td>-indexrangesize n</td>
<td>Specifies the number of indexes to track for access statistics</td>
</tr>
<tr>
<td>Table Range Size</td>
<td>-tablerangesize n</td>
<td>Specifies the number of tables for which you want to collect statistics</td>
</tr>
</tbody>
</table>
Database server consistency check parameters

Use the parameters listed in Table 19–5 to enable consistency checking.

Table 19–5: Server consistency check parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area consistency</td>
<td>-AreaCheck area-name</td>
<td>Area consistency check enables consistency checking for all record and index blocks in the specified area</td>
</tr>
<tr>
<td>check</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database consistency check</td>
<td>-DbCheck</td>
<td>Database consistency check enables consistency checking for all record and index blocks in the entire database</td>
</tr>
<tr>
<td>Memory consistency check</td>
<td>-MemCheck</td>
<td>Memory consistency check enable memory overwrite checking for the buffer pool</td>
</tr>
<tr>
<td>Index consistency check</td>
<td>-IndexCheck index-name</td>
<td>Index consistency check enables consistency checking for all index blocks in the specified index</td>
</tr>
<tr>
<td>Table consistency</td>
<td>-TableCheck table-name</td>
<td>Table consistency check enables consistency checking for all record blocks of the specified table (excluding BLOBS)</td>
</tr>
</tbody>
</table>
Database server SSL connection parameters

Use the parameters listed in Table 19–6 to control the use of SSL.

Table 19–6: Database server SSL parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSL</td>
<td>-ssl</td>
<td>Requires that all brokers and all connections use SSL</td>
</tr>
<tr>
<td></td>
<td></td>
<td><strong>Note</strong>: SSL incurs heavy performance penalties, depending on the client, server, and network resources and load.</td>
</tr>
<tr>
<td>Key Alias</td>
<td>-keyalias key-alias-name</td>
<td>Specifies the alias name of the SSL private key/digital certificate key-store entry to use</td>
</tr>
<tr>
<td>Key Alias Password</td>
<td>-keyaliaspasswd key-alias-password</td>
<td>Specifies the encrypted password to the key alias</td>
</tr>
<tr>
<td>No Session Cache</td>
<td>-nosessioncache</td>
<td>Disables the use of SSL session caching</td>
</tr>
<tr>
<td>Session Timeout</td>
<td>-sessiontimeout n</td>
<td>Specifies the length of time in seconds that an SSL session is held in the session cache</td>
</tr>
</tbody>
</table>
Database server network parameters

Use the parameters listed in Table 19–7 to supply the broker with necessary network information.

Table 19–7: Server network parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Syntax</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>AdminServer Port</td>
<td>-adminport { service-name</td>
<td>port }</td>
</tr>
<tr>
<td>SQL Server Java™ Classpath</td>
<td>-classpath pathname</td>
<td>Identifies the Java™ classpath to use when starting an SQL server</td>
</tr>
<tr>
<td>Host Name</td>
<td>-H host-name</td>
<td>Specifies a remote host</td>
</tr>
<tr>
<td>Maximum Clients Per Server</td>
<td>-Ma n</td>
<td>Specifies the maximum number of remote users per database server</td>
</tr>
<tr>
<td>Maximum Dynamic Server</td>
<td>-maxport n</td>
<td>Specifies the highest accessible port number in a specified range</td>
</tr>
<tr>
<td>Minimum Clients Per Server</td>
<td>-Mi n</td>
<td>Specifies the number of remote users on a server before a broker starts another server</td>
</tr>
<tr>
<td>Minimum Dynamic Server</td>
<td>-minport n</td>
<td>Specifies the lowest accessible port number in a specified range</td>
</tr>
<tr>
<td>Maximum Servers</td>
<td>-Mn n</td>
<td>Specifies the maximum number of remote client servers that a broker can start</td>
</tr>
<tr>
<td>Servers Per Protocol</td>
<td>-Mp n</td>
<td>Specifies the maximum number of servers to serve remote users for a protocol</td>
</tr>
<tr>
<td>Maximum Servers Per Broker</td>
<td>-Mpb n</td>
<td>Specifies the maximum number of servers that multiple brokers can start to serve remote users for a protocol</td>
</tr>
<tr>
<td>Network Type</td>
<td>-N network-type</td>
<td>Identifies the network communications protocol</td>
</tr>
<tr>
<td>Pending Connection Time</td>
<td>-PendConnTime n</td>
<td>Specifies the amount of time a client has to connect to a server before the broker clears the client’s reservation</td>
</tr>
<tr>
<td>Parameter</td>
<td>Syntax</td>
<td>Purpose</td>
</tr>
<tr>
<td>---------------------------</td>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Configuration Properties File</td>
<td>-properties filename</td>
<td>Identifies the properties file an AdminServer uses when starting a database server or server group</td>
</tr>
<tr>
<td>Service Name</td>
<td>-s { service-name</td>
<td>port-number }</td>
</tr>
<tr>
<td>Server Group</td>
<td>-servergroup name</td>
<td>Identifies a logical collection of server processes to start</td>
</tr>
<tr>
<td>Century Year Offset</td>
<td>-yy n</td>
<td>Determines the start of the 100-year period in which a date with a two-digit year is valid</td>
</tr>
</tbody>
</table>
Startup parameter usage categories

Startup parameters are organized into usage categories. Table 19–8 describes each usage category.

Table 19–8:  Startup parameter categories

<table>
<thead>
<tr>
<th>Usage type</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Session (CS)</td>
<td>Starts a client session</td>
</tr>
<tr>
<td>Client Connections (CC)</td>
<td>Connects to a specific database</td>
</tr>
<tr>
<td>OpenEdge Database Server (DBS)</td>
<td>Starts a server or broker</td>
</tr>
<tr>
<td>DataServer (DS)</td>
<td>Starts non-OpenEdge DataServers</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-adminport service-name port</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Use with

DBS

Maximum value

Minimum value

Single-user default

Multi-user default

**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 7832.

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-aiarcdir dirlist</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-aiarcdircreate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)

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.

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-aiarcinterval ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>86400</td>
<td>120(^1)</td>
</tr>
</tbody>
</table>

1. For timed mode, the minimum value is 120. For on-demand mode, do not specify this parameter.

\( n \) 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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-aibufs n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>1</td>
<td>–</td>
</tr>
</tbody>
</table>

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 Chapter 21, “PROUTIL Utility”.

After-image Stall (-aistall)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-aistall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-AreaCheck areaname</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CC, DBS</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

Specify 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).

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” section on page 20–88 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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th><code>-B n</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CC, DBS</td>
<td>1,000,000,000  or 125,000,000²</td>
<td>10</td>
</tr>
</tbody>
</table>

¹. The *users* value is specified by the Number of Users (`-n`) parameter.

². 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.

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. Starting in Release 10.1C, the OpenEdge RDBMS expands the range of internal validations used to ensure database consistency in both index and data blocks during forward processing. Validations using `-B` can be run online as part of routine health checks. For more information, see Chapter 21, “PROUTIL Utility”.

**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*. 

---

19–18
# Blocks in Alternate Buffer Pool (-B2)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-B2 n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max.</td>
<td>Min.</td>
<td>Sing.</td>
</tr>
<tr>
<td>CC, DBS</td>
<td>1,000,000,000 or 125,000,000²</td>
<td>10</td>
<td>0</td>
</tr>
</tbody>
</table>

1. The **users** value is specified by the Number of Users (-n) parameter.
2. 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.

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.
**Base Index (-baseindex)**

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-baseindex n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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:

```abl
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:

<table>
<thead>
<tr>
<th>File-Name</th>
<th>filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>_idx-num</td>
<td>Index-Name</td>
</tr>
<tr>
<td>n1</td>
<td>index name1</td>
</tr>
<tr>
<td>n2</td>
<td>index name2</td>
</tr>
<tr>
<td>n3</td>
<td>index name3</td>
</tr>
</tbody>
</table>
Base Table (-basetable)

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
```
### Before-image Buffers (-bibufs)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-bibufs n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>3</td>
<td>–</td>
</tr>
</tbody>
</table>

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 Chapter 21, “PROUTIL Utility”.

### Threshold Stall (-bistall)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-bistall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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)

Recovery Log Threshold (-bithold)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-bithold n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>System dependent¹</td>
<td>System dependent¹</td>
</tr>
</tbody>
</table>

¹. Limited by available disk space.

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 -bista11 parameter with -bithold prevents the emergency shutdown; instead, the database ceases activity until the BI extent is emptied.
Schema Cache File (-cache)

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.

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.
### SQL Server Java Classpath (-classpath)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-classpath <code>pathname</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

**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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-cluster qualifier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Cluster Mode (-cluster)

Direct a broker to start a cluster-enabled database directly, or to instruct the cluster manager to perform the startup. Table 19–9 describes the accepted values for qualifier.

Table 19–9: Values for -cluster

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>startup</td>
<td>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.</td>
</tr>
<tr>
<td>protected</td>
<td>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. <strong>Note:</strong> This parameter must only be used by the cluster manager. It should not be used on the command line or a directly executed script.</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-convmap filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CS, DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-cpcase tablename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CS, DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-cpcoll tablename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>CS, DBS</td>
<td>–</td>
<td>–</td>
<td>Basic</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>-cpinternal codepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CS, DBS, DS</td>
<td>iso8859-1</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>-cplog codepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CS, DBS, DS</td>
<td>-cpstream</td>
</tr>
</tbody>
</table>

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)**

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-cpprint codepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CS, DBS, DS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

*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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CS, DBS, DS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

`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)

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 (-cpter m)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-cpter m codepage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>CS, DBS, DS</td>
<td>–</td>
<td>–</td>
<td>-cpstream</td>
</tr>
</tbody>
</table>

codepage

The name of the code page for character terminals.

Use Terminal Code Page (-cpter m) 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 -cpter m.

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.
Database consistency check (-DbCheck)

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), one table with Table consistency check (-TableCheck), or one index with Index consistency check (-IndexCheck).

Database consistency checking can be enabled or disabled while your database is online with PROMON. See the “Enable/disable block level consistency check” section on page 20–88 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.

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-DbCheck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CC, DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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), one table with Table consistency check (-TableCheck), or one index with Index consistency check (-IndexCheck).

Database consistency checking can be enabled or disabled while your database is online with PROMON. See the “Enable/disable block level consistency check” section on page 20–88 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)

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)

For Enterprise licenses only, use Encryption Cache Size (-ecsize) to specify the size (in number of encryption policies) of the caches for encryption policies. 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 Chapter 10, “Transparent Data Encryption,” for more information on transparent data encryption.

### Event Level (-evtlevel)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>Windows</th>
<th>-evtlevel value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS, CS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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 Chapter 17, “Logged Data.”
Before-image Truncate Interval (-G)

-\( n \)

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.

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-G ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>32</td>
<td>0</td>
</tr>
</tbody>
</table>

Group Delay (-groupdelay)

-\( n \)

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)

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)

The number of hash table entries to use for the buffer pool.

Caution: Do not use the -hash parameter unless directed to do so by Technical Support.
No Crash Protection (-i)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CC, DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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 Chapter 6, “Recovering a Database.”
Index consistency check (-IndexCheck)

owner-name

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, or the table is owned by PUB. By default, ABL tables are owned by PUB.

table-name

Specifies the table containing the index to be checked.

indexname

Specifies the name 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 with -IndexCheck. To check multiple indexes, you can apply consistency checking to the entire database with Database consistency check (-DbCheck), or if multiple indexes are in one area, you can apply consistency checking to an entire area with Area consistency check (-AreaCheck).

Index consistency checking can be enabled or disabled while your database is online with PROMON. See the “Enable/disable block level consistency check” section on page 20–88 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)

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 Chapter 26, “Virtual System Tables,” for more information on VSTs.

Internet Protocol (-ipver)

Specifies Internet Protocol Version 4. Only network connections with IPv4 are allowed.


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.

Use Internet Protocol (-ipver) to identify the IP version for network connections.
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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-keyalias key-alias-name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

*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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-keyaliaspasswd key-alias-password</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

1. The actual value used is the encrypted value of the string “password”.

**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 Table Entries (-L)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-L n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>System dependent(^1)</td>
<td>32</td>
</tr>
</tbody>
</table>

1. Limited by available memory.

**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 Chapter 21, “PRUTIL Utility”.

### Lock Table Hash Size (-lkhash)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-lkhash n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

1. The default is variable based on the size of your lock table (-l).

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)

Lock Release (-lkrela) uses the original lock release mechanism installed from previous OpenEdge versions.

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-lkrela</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>Client Session</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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.
Auto Server (-m1)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-m1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max. value</td>
<td>Min. value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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.

Manual Server (-m2)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-m2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max. value</td>
<td>Min. value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Use Manual Server (-m2) to manually start a remote user server after you start a broker (servers are generally started automatically by the broker process). Use this parameter in the following cases:

- For debugging purposes, to start servers directly and observe their behavior
- On systems where automatic server generation is not possible

Secondary Login Broker (-m3)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-m3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Max. value</td>
<td>Min. value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

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.
Database Startup Parameters

Maximum Clients per Server (-Ma)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-Ma n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>2048</td>
<td>1</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-maxAreas n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>32000</td>
<td>Variable(^1)</td>
</tr>
</tbody>
</table>

\(^1\) The minimum value is the current maximum area number in use for the database.

n

The maximum area number.

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 Chapter 15, “Maintaining Database Structure.”

**Maximum Dynamic Server (-maxport)**

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-maxport n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>System dependent</td>
<td>System dependent</td>
<td>–</td>
</tr>
</tbody>
</table>

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)**

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-maxxids n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>32,000</td>
<td>0, (users)/2</td>
<td>–</td>
</tr>
</tbody>
</table>

1. For a JTA-enabled database, the minimum value is (users)/2 where users is the value of the -n startup parameter. For a database not enabled for JTA, the minimum is zero (0).

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) thus enabling the database to function as a resource manager as defined in the J2EE framework architecture. Because the transactions are controlled by an external transaction manager, they may take a long time to complete. Use the Maximum JTA Transactions parameter (-maxxids) to control the number of JTA transactions simultaneously allowed.
Memory consistency check (-MemCheck)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-MemCheck</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CC, DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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 the “Enable/disable block level consistency check” section on page 20–88 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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-Mf n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>32,768</td>
<td>0</td>
<td>0^1</td>
</tr>
</tbody>
</table>

1. Default is 3 for batch jobs.

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).

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.

Operating system and syntax

UNIX

Windows

- Mf n

3

n

DBS

32,768

0

0^1

3
Minimum Clients per Server (-Mi)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-Mi n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>1</td>
</tr>
</tbody>
</table>

The number of remote users on a server before the broker starts another server. See the “Maximum Servers (-Mn)” section on page 19–51 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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-minport n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>1,025</td>
</tr>
</tbody>
</table>

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.
Maximum Servers (-Mn)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-Mn n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>512</td>
<td>1</td>
<td>–</td>
</tr>
</tbody>
</table>

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. This parameter applies only to databases that are accessed from remote network nodes. 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 the “Maximum Clients per Server (-Ma)” section on page 19–46 and the “Minimum Clients per Server (-Mi)” section on page 19–50 for more information.

Servers per Protocol (-Mp)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-Mp n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>Value of -Mn</td>
<td>1</td>
<td>–</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-Mpb n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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.

### VLM Page Table Entry Optimization (-Mpte)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>Digital UNIX</th>
<th>-Mpte</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

Use VLM Page Table Entry Optimization (-Mpte) to allocate shared memory in multiples of 8MB at server startup for VLM64 support. This function is a binary switch that is off by default. The -Mpte startup parameter turns on the function.

### User MUX Latches (-mux)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-mux n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>1</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>CC</td>
<td>1</td>
<td>0</td>
<td>–</td>
</tr>
</tbody>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th><code>-Mxs n</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
</tbody>
</table>
| DBS                         | System dependent\(^1\) | 1 | – | For 32-bit platforms: \(16384 + (\#\ of\ users \* 300) / 1024\)  
For 64-bit platforms: \(16384 + (\#\ of\ users \* 400) / 1024\) |

1. The maximum is limited only by the size of the signed integer data type on the system.

\(n\)

The size of the shared-memory overflow area in kilobytes.

Use Shared-memory Overflow (\(-\text{Mxs}\)) to replace the default value of the shared-memory overflow area; it does not increase it. 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 \(-\text{Mxs}\) can be run online as part of routine health checks. For more information, see Chapter 21, “PROUTIL Utility”.

**Specific memory requirements when using mandatory fields**

If you are using mandatory fields, your approximate memory requirements should consider the following when calculating the value of Shared Memory Overflow (\(-\text{Mxs}\)):

- 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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-N network-type</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use with</strong></td>
<td><strong>Maximum value</strong></td>
<td><strong>Minimum value</strong></td>
<td><strong>Single-user default</strong></td>
</tr>
<tr>
<td>CC, DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

*network-type*

The network communications protocol.

Use Network Type (-N) to specify the network communications protocol supported by the OpenEdge RDBMS. TCP is the only supported option.

### Number of Users (-n)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-n n</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Use with</strong></td>
<td><strong>Maximum value</strong></td>
<td><strong>Minimum value</strong></td>
<td><strong>Single-user default</strong></td>
</tr>
<tr>
<td>DBS</td>
<td>10,000</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

*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), PROWDODG processes, and PROMON sessions.

See the “Maximum Clients per Server (-Ma)” section on page 19–46 and the “Maximum Servers (-Mn)” section on page 19–51 for more information.
Nap Time Increment (-napinc)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-napinc n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>10</td>
</tr>
</tbody>
</table>

Use Nap Time Increment (-napinc) to specify the nap time increment for Enterprise database licenses only.

Do not use this parameter unless directed to do so by Progress Software Technical Support.

Nap Time Steps (-napstep)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-napstep n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>5</td>
</tr>
</tbody>
</table>

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.

No Session Cache (-nosessioncache)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-nosessioncache</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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.
Storage Object Cache Size (-omsize)

Use Storage Object Cache Size (-omsize) to specify the size of the object cache for all database objects.

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-omsize n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>65535</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>CC</td>
<td>–</td>
<td>–</td>
<td>1024</td>
</tr>
</tbody>
</table>

Prompt for Passphrase (-Passphrase)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-Passphrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td>CC</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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 the Chapter 10, “Transparent Data Encryption.”
Pending Connection Time (-PendConnTime)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-PendConnTime n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>

\( n \)

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-pf filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CC, CS, DBS, DS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

\( filename \)

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-pica n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| DBS | 8192 | 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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-pinshm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| 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.

### Configuration Properties File (-properties)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-properties filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>filename</th>
</tr>
</thead>
</table>

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-r</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>CC</td>
<td>–</td>
<td>–</td>
<td>Unbuffered I/O</td>
</tr>
</tbody>
</table>

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.

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 Progress 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 the “Server Group (-servergroup)” section on page 19–61 for more information.

Table 19–10 shows how the broker, server, and remote client interpret each of their parameters when you use the -S parameter.

### Table 19–10: Parameter interpretation with service name (-S)

<table>
<thead>
<tr>
<th>Module</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>Parameters apply to the connections on which the broker is listening for connection requests from remote clients</td>
</tr>
<tr>
<td>Server</td>
<td>Parameters apply to the connection between the server and the remote client</td>
</tr>
<tr>
<td>Remote Client</td>
<td>Parameters identify the connection parameters to the broker or the server</td>
</tr>
</tbody>
</table>

To run a multi-user OpenEdge session from a remote network node, use both the Host Name (-H) and Service Name (-S) parameters.

### Semaphore Sets (-semsets)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>-semsets ( n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>Maximum Number of Users + 1</td>
<td>1</td>
</tr>
</tbody>
</table>

\( n \)

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)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-servergroup name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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 Explorer or Progress 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 Explorer, Progress Explorer, or the DBMAN utility.

Session Timeout (-sessiontimeout)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-sessiontimeout n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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)

The maximum number of bytes, specified in MB or GB, in a shared memory segment. Table 19–11 lists the supported values by platform.

Table 19–11: Values for -shmsegsize

<table>
<thead>
<tr>
<th>Platform</th>
<th>Values in MB</th>
<th>Values in GB¹</th>
</tr>
</thead>
<tbody>
<tr>
<td>32-bit platforms</td>
<td>128</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>512</td>
<td>–</td>
</tr>
<tr>
<td></td>
<td>1024</td>
<td>1g</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>2g</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>4g</td>
</tr>
<tr>
<td>64-bit platforms</td>
<td>1024</td>
<td>1g</td>
</tr>
<tr>
<td></td>
<td>2048</td>
<td>2g</td>
</tr>
<tr>
<td></td>
<td>4096</td>
<td>4g</td>
</tr>
<tr>
<td></td>
<td>8192</td>
<td>8g</td>
</tr>
<tr>
<td></td>
<td>16384</td>
<td>16g</td>
</tr>
<tr>
<td></td>
<td>32768</td>
<td>32g</td>
</tr>
</tbody>
</table>

1. 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 the “Shared memory allocation” section on page 14–27 for more information.

1. Limited by available memory.

2. 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.
Alphabetical listing of database startup parameters

**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`)

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th><code>-spin n</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>0</td>
<td>–</td>
</tr>
</tbody>
</table>

\(^1\) For single CPU systems, the default is 10,000. For multi-CPU systems, the default is 6000 * # of CPUs.

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 the “PROMON R&D Advanced Options” section on page 20–24. 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.
### SSL (-ssl)

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.*

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-ssl</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-TableCheck [owner-name.] tablename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td></td>
<td></td>
<td>-TableCheck [owner-name.] tablename</td>
</tr>
<tr>
<td>Maximum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Minimum value</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Single-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Multi-user default</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*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.

*tablename*

Specifies the name of the table for consistency checking.

Use Table consistency check to enable consistency checking for all record blocks of the specified table (excluding BLOBS). Table consistency check validates a block is still physically correct after an operation has been performed.

You can only specify one table with -TableCheck. To check multiple tables, you must apply consistency checking to the entire database with Database consistency check (-DbCheck) or if multiple tables are in one area, you can apply consistency checking to an entire area with Area consistency check (-AreaCheck).
Table consistency checking can be enabled or disabled while your database is online with PROMON. See the “Enable/disable block level consistency check” section on page 20–88 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)**

<table>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX</th>
<th>Windows</th>
<th>-tablerangesize n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
<td>Single-user default</td>
</tr>
<tr>
<td>DBS</td>
<td>–</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

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.
Century Year Offset (-yy)

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>
<thead>
<tr>
<th>Operating system and syntax</th>
<th>UNIX Windows</th>
<th>-yy n</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use with</td>
<td>Maximum value</td>
<td>Minimum value</td>
</tr>
<tr>
<td>CS, DBS</td>
<td>–</td>
<td>–</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>-yy</th>
<th>Year (as supplied in DATE format)</th>
<th>Result of year function</th>
</tr>
</thead>
<tbody>
<tr>
<td>1900</td>
<td>50 to 99</td>
<td>1950 to 1999</td>
</tr>
<tr>
<td></td>
<td>00 to 49</td>
<td>1900 to 1949</td>
</tr>
<tr>
<td>1950</td>
<td>50 to 99</td>
<td>1950 to 1999</td>
</tr>
<tr>
<td></td>
<td>00 to 49</td>
<td>2000 to 2049</td>
</tr>
<tr>
<td>1980</td>
<td>80 to 99</td>
<td>1980 to 1999</td>
</tr>
<tr>
<td></td>
<td>00 to 79</td>
<td>2000 to 2079</td>
</tr>
</tbody>
</table>

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.
This chapter describes the OpenEdge database administration utility PROMON.
PROMON utility

Starts the monitor that displays database information.

**Syntax**

```
promon db-name
```

**Parameters**

`db-name`

Specifies the database you want to monitor.

```
-userid username [-password passwd]
```

Specifies the userid and password of an authenticated database administrator.

When you enter the PROMON utility, the main menu appears, as shown in Figure 20–1.

---

OpenEdge MONITOR Version 10

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

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:

---

**Figure 20–1: 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:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Display all entries</td>
</tr>
<tr>
<td>2.</td>
<td>Match a user number</td>
</tr>
<tr>
<td>3.</td>
<td>Match a range of user numbers</td>
</tr>
<tr>
<td>Q.</td>
<td>Return to main menu</td>
</tr>
</tbody>
</table>

Enter your selection:

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.
PROMON User Control option

Displays information for each database process. Figure 20–2 shows an example of this option’s output.

Table 20–1: Process type values

<table>
<thead>
<tr>
<th>Value</th>
<th>Process type</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIMD</td>
<td>After-image file management utility daemon</td>
</tr>
<tr>
<td>AIW</td>
<td>After-image writer (AIW) process</td>
</tr>
<tr>
<td>APW</td>
<td>Asynchronous page writer</td>
</tr>
<tr>
<td>AUDA</td>
<td>Audit archive utility</td>
</tr>
<tr>
<td>AUDL</td>
<td>Audit load utility</td>
</tr>
<tr>
<td>BAT</td>
<td>Batch user</td>
</tr>
<tr>
<td>BIND</td>
<td>Binary dump utility</td>
</tr>
<tr>
<td>BINL</td>
<td>Binary load utility</td>
</tr>
<tr>
<td>BIW</td>
<td>Before-image writer</td>
</tr>
<tr>
<td>BKUP</td>
<td>Online backup</td>
</tr>
<tr>
<td>BROK</td>
<td>Broker process</td>
</tr>
<tr>
<td>FMA</td>
<td>OpenEdge Management (formerly Fathom™ Management) Monitor</td>
</tr>
</tbody>
</table>

Figure 20–2: Sample output for PROMON User Control option

**Usr**

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.

**Type**

Table 20–1 lists the distinct types of database processes that can appear in the Type field.
Table 20–1: Process type values

<table>
<thead>
<tr>
<th>Value</th>
<th>Process type</th>
</tr>
</thead>
<tbody>
<tr>
<td>IACT</td>
<td>Index activate utility</td>
</tr>
<tr>
<td>IDX C</td>
<td>Index compact utility</td>
</tr>
<tr>
<td>IDX F</td>
<td>Index fix utility</td>
</tr>
<tr>
<td>IDX M</td>
<td>Index move utility</td>
</tr>
<tr>
<td>MON</td>
<td>Monitor process</td>
</tr>
<tr>
<td>PSA</td>
<td>PROSTRCT ADD utility</td>
</tr>
<tr>
<td>QUIE</td>
<td>PROQUIET utility</td>
</tr>
<tr>
<td>REMC</td>
<td>Remote client</td>
</tr>
<tr>
<td>RFUT</td>
<td>RFUTIL utility</td>
</tr>
<tr>
<td>RPLA</td>
<td>OpenEdge Replication Site Replication Agent</td>
</tr>
<tr>
<td>RPLE</td>
<td>OpenEdge Replication Site Replication online enable</td>
</tr>
<tr>
<td>RPLS</td>
<td>OpenEdge Replication Site Replication Server</td>
</tr>
<tr>
<td>RPLT</td>
<td>OpenEdge Replication Site Replication Transition</td>
</tr>
<tr>
<td>RPLU</td>
<td>OpenEdge Replication Site Replication Utilities</td>
</tr>
<tr>
<td>SELF</td>
<td>Self-service client</td>
</tr>
<tr>
<td>SERV</td>
<td>Remote-user server</td>
</tr>
<tr>
<td>SHUT</td>
<td>Database shutdown process</td>
</tr>
<tr>
<td>SQSC</td>
<td>SQLSCHEMA utility</td>
</tr>
<tr>
<td>SQDP</td>
<td>SQLDUMP utility</td>
</tr>
<tr>
<td>SQLD</td>
<td>SQLLOAD utility</td>
</tr>
<tr>
<td>SQSV</td>
<td>SQL server</td>
</tr>
<tr>
<td>TBLM</td>
<td>Table move utility</td>
</tr>
<tr>
<td>WDOG</td>
<td>Watchdog</td>
</tr>
</tbody>
</table>
Wait

Indicates that a process is waiting for a lock or latch. Table 20–2 lists common values and describes the resource or event associated with each value.

**Table 20–2: Possible wait column values**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Lock waits</strong></td>
<td></td>
</tr>
<tr>
<td>REC</td>
<td>Row (record) lock</td>
</tr>
<tr>
<td>IX</td>
<td>Index lock</td>
</tr>
<tr>
<td>SCHE</td>
<td>Schema lock</td>
</tr>
<tr>
<td>TRAN</td>
<td>Transaction commit</td>
</tr>
<tr>
<td>TXB</td>
<td>Transaction end update mode lock</td>
</tr>
<tr>
<td>TXE</td>
<td>Transaction end commit mode lock</td>
</tr>
<tr>
<td>TXS</td>
<td>Transaction end share lock</td>
</tr>
<tr>
<td>TXX</td>
<td>Transaction end exclusive lock</td>
</tr>
<tr>
<td><strong>Resource waits</strong></td>
<td></td>
</tr>
<tr>
<td>AIRD</td>
<td>After-image buffer read from disk</td>
</tr>
<tr>
<td>AIWR</td>
<td>After-image buffer write to disk</td>
</tr>
<tr>
<td>BIRD</td>
<td>Before-image buffer read from disk</td>
</tr>
<tr>
<td>BIWR</td>
<td>Before-image buffer write to disk</td>
</tr>
<tr>
<td>BKEX</td>
<td>Exclusive lock on database buffer</td>
</tr>
<tr>
<td>BKSH</td>
<td>Share lock on database buffer</td>
</tr>
<tr>
<td>BKSI</td>
<td>Share lock with intent to update on database buffer</td>
</tr>
<tr>
<td>BUFF</td>
<td>Shared buffer</td>
</tr>
<tr>
<td>DB</td>
<td>Database server</td>
</tr>
<tr>
<td>DBBK</td>
<td>Database buffer being backed up</td>
</tr>
<tr>
<td>DBRD</td>
<td>Database buffer read from disk</td>
</tr>
<tr>
<td>DBWR</td>
<td>Database buffer write to disk</td>
</tr>
<tr>
<td>DEAD</td>
<td>Process is waiting to be removed from the process list¹</td>
</tr>
<tr>
<td>RGET</td>
<td>Row get lock</td>
</tr>
<tr>
<td>SRPL</td>
<td>Replication agent acknowledgement</td>
</tr>
</tbody>
</table>

¹ This notation is not explained in the text, but it likely refers to the current status of the process within the system.
You may also see the following values listed in the **Wait** column:

- BF2
- BF3
- BF4
- BIB

### Table 20–2: Possible wait column values

<table>
<thead>
<tr>
<th>Latch waits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIB</td>
<td>After-image buffer latch</td>
</tr>
<tr>
<td>BFP</td>
<td>Buffer pool latch</td>
</tr>
<tr>
<td>BHT</td>
<td>Buffer pool hash table latch</td>
</tr>
<tr>
<td>BUF</td>
<td>Buffer pool buffer latch</td>
</tr>
<tr>
<td>CPQ</td>
<td>Checkpoint queue latch</td>
</tr>
<tr>
<td>DLC</td>
<td>Index delete chain</td>
</tr>
<tr>
<td>GST</td>
<td>Global storage pool latch</td>
</tr>
<tr>
<td>IXD</td>
<td>Index delete chain latch</td>
</tr>
<tr>
<td>LKF</td>
<td>Lock table free chain latch</td>
</tr>
<tr>
<td>LKP</td>
<td>Lock table purge chain latch</td>
</tr>
<tr>
<td>LKT</td>
<td>Lock table latch</td>
</tr>
<tr>
<td>LRS</td>
<td>Secondary LRU chain (not currently implemented)</td>
</tr>
<tr>
<td>LRU</td>
<td>Buffer pool LRU chain latch</td>
</tr>
<tr>
<td>MTX</td>
<td>Log space allocation latch</td>
</tr>
<tr>
<td>PWQ</td>
<td>Page writer queue latch</td>
</tr>
<tr>
<td>SCH</td>
<td>Shared schema cache latch</td>
</tr>
<tr>
<td>SEQ</td>
<td>Sequence cache latch</td>
</tr>
<tr>
<td>TXQ</td>
<td>Transaction queue latch</td>
</tr>
<tr>
<td>TXT</td>
<td>Transaction table latch</td>
</tr>
<tr>
<td>USR</td>
<td>User table latch</td>
</tr>
</tbody>
</table>

1. Seen when a process is terminated with the UNIX `kill-9` command
PROMON User Control option

- LRI
- LRX
- OM

The previous values are for short-duration (micro- to milliseconds) latches used internally by the OpenEdge RDBMS.

Area Dbkey

If the user has a REC wait, the combination of the area number and the db key, identifies the record.

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.
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. Figure 20–3 shows an example of this option’s output.

<table>
<thead>
<tr>
<th>Type</th>
<th>Usr</th>
<th>Name</th>
<th>Record</th>
<th>Trans</th>
<th>Schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lock 999 TOTAL...</td>
<td>2345</td>
<td>567</td>
<td>145</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 999 TOTAL...</td>
<td>89</td>
<td>12</td>
<td>9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock 0</td>
<td>654</td>
<td>32</td>
<td>33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock 1 sue</td>
<td>54</td>
<td>6</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 1 sue</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock 2 dave</td>
<td>43</td>
<td>7</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 2 dave</td>
<td>21</td>
<td>3</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock 3</td>
<td>1123</td>
<td>302</td>
<td>54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 3</td>
<td>12</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock 4 mary</td>
<td>345</td>
<td>89</td>
<td>30</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 4 mary</td>
<td>20</td>
<td>6</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock 5 bill</td>
<td>101</td>
<td>20</td>
<td>15</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 5 bill</td>
<td>10</td>
<td>2</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock 6</td>
<td>125</td>
<td>34</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Wait 6</td>
<td>15</td>
<td>3</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RETURN - show remaining, U - continue uninterrupted, Q - quit:

**Figure 20–3: Sample output for PROMON Locking and Waiting statistics option**

**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**

The user number of the process. User number 999 represents all users.

**Name**

For client processes, the user 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. Figure 20–4 shows an example of this option’s output.

<table>
<thead>
<tr>
<th>Type</th>
<th>Usr</th>
<th>Name</th>
<th>DB Requests</th>
<th>DB Reads</th>
<th>BI Reads</th>
<th>AI Reads</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc</td>
<td>999</td>
<td>TOTAL</td>
<td>17666</td>
<td>472</td>
<td>198</td>
<td>0</td>
</tr>
<tr>
<td>Acc</td>
<td>0</td>
<td>george</td>
<td>8553</td>
<td>388</td>
<td>198</td>
<td>0</td>
</tr>
<tr>
<td>Acc</td>
<td>5</td>
<td>bill</td>
<td>946</td>
<td>59</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Acc</td>
<td>6</td>
<td>sue</td>
<td>918</td>
<td>6</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Acc</td>
<td>7</td>
<td>mary</td>
<td>909</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Acc</td>
<td>8</td>
<td>erin</td>
<td>921</td>
<td>18</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

RETURN - repeat, U - continue uninterrupted, Q - quit:

Figure 20–4: Sample output for PROMON Block Access option

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

The user number of the process.

Name

For client processes, the user name.

DB Reqst

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. Figure 20–5 shows an example of this option’s output.

<table>
<thead>
<tr>
<th>Record Locking Table:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usr Name</td>
</tr>
<tr>
<td>------------------------</td>
</tr>
<tr>
<td>6 shannon REC 120</td>
</tr>
<tr>
<td>7 shannon REC 120</td>
</tr>
<tr>
<td>8 shannon REC 120</td>
</tr>
</tbody>
</table>

RETURN - repeat, U - continue uninterrupted, Q - quit

Figure 20–5: Sample output for PROMON Record Locking Table option

The size of the record locking table is set with the Locking Table Entries (-L) startup parameter. See the chapter on locks in OpenEdge Getting Started: ABL Essentials for more information on locks and locking.

Usr

The user number.

Name

For client processes, the user 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.

Rec-id  

The record ID for the lock table entry. The Rec-id column identifies the records locked by each database process.

Table  

The 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. Table 20–3 lists the flags and their meanings.

Table 20–3: Flag values

<table>
<thead>
<tr>
<th>Flag</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>Limbo lock</td>
<td>The client has released the record, but the transaction has not completed. The record lock is not released until the transaction ends.</td>
</tr>
<tr>
<td>P</td>
<td>Purged lock entry</td>
<td>The lock is no longer held.</td>
</tr>
<tr>
<td>Q</td>
<td>Queued lock request</td>
<td>Represents a queued request for a lock already held by another process.</td>
</tr>
<tr>
<td>U</td>
<td>Upgrade request</td>
<td>The user has requested a lock upgrade from SHARE to EXCLUSIVE but is waiting for another process that holds a conflicting lock.</td>
</tr>
<tr>
<td>H</td>
<td>No hold</td>
<td>The “nohold” flag is set.</td>
</tr>
</tbody>
</table>

Trans State

The state of the transaction. Table 20–4 lists the possible states.

Table 20–4: Transaction states

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Begin</td>
<td>A transaction table entry was allocated, and a start record is being logged</td>
</tr>
<tr>
<td>Active</td>
<td>The transaction is doing forward processing</td>
</tr>
<tr>
<td>Dead</td>
<td>The transaction is complete, but the lock has not been released</td>
</tr>
<tr>
<td>Prep</td>
<td>The transaction is preparing to enter phase 1 (ready to commit), but has not sent a ready-to-commit reply</td>
</tr>
<tr>
<td>Phase 1</td>
<td>In phase 1, ready to commit</td>
</tr>
<tr>
<td>Phase 2</td>
<td>In phase 2</td>
</tr>
<tr>
<td>C</td>
<td>With two-phase commit, this user is logging for the coordinator for the transaction</td>
</tr>
<tr>
<td>R</td>
<td>Ready to commit</td>
</tr>
<tr>
<td>L</td>
<td>Limbo transaction</td>
</tr>
<tr>
<td>Active JTA</td>
<td>The transaction is currently executing</td>
</tr>
<tr>
<td>Idle JTA</td>
<td>The transaction is not currently executing</td>
</tr>
</tbody>
</table>
### Table 20–4: Transaction states (2 of 2)

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepared JTA</td>
<td>The transaction is prepared</td>
</tr>
<tr>
<td>RollbackOnly JTA</td>
<td>The transaction has encountered an error</td>
</tr>
<tr>
<td>Committed JTA</td>
<td>The transaction is in the commit process</td>
</tr>
</tbody>
</table>

**Trans ID**

Transaction ID of the lock.
PROMON Activity option

Displays an overview of your system activity. Figure 20–6 shows an example of this option’s output.

<table>
<thead>
<tr>
<th>Event</th>
<th>Total</th>
<th>Per Sec</th>
<th>Event</th>
<th>Total</th>
<th>Per Sec</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commits</td>
<td>50</td>
<td>0.2</td>
<td>Undos</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Updates</td>
<td>0</td>
<td>0.0</td>
<td>Record Reads</td>
<td>3709</td>
<td>16.9</td>
</tr>
<tr>
<td>Record Creates</td>
<td>0</td>
<td>0.0</td>
<td>Record Deletes</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>DB Writes</td>
<td>3</td>
<td>0.0</td>
<td>DB Read</td>
<td>473</td>
<td>2.2</td>
</tr>
<tr>
<td>BI Writes</td>
<td>2</td>
<td>0.0</td>
<td>BI Reads</td>
<td>198</td>
<td>0.9</td>
</tr>
<tr>
<td>AI Writes</td>
<td>0</td>
<td>0.0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Record Locks</td>
<td>210</td>
<td>1.0</td>
<td>Record Waits</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Checkpoints</td>
<td>0</td>
<td>0.0</td>
<td>Buffs Flushed</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Rec Lock Waits</td>
<td>0 %</td>
<td>BI Buf Waits</td>
<td>0 %</td>
<td>AI Buf Waits</td>
<td>0 %</td>
</tr>
<tr>
<td>Writes by APW</td>
<td>0 %</td>
<td>Writes by BIW</td>
<td>0 %</td>
<td>Writes by AIW</td>
<td>0 %</td>
</tr>
<tr>
<td>Buffer Hits</td>
<td>99 %</td>
<td>Primary Hits</td>
<td>100 %</td>
<td>Alternate Hits</td>
<td>99 %</td>
</tr>
<tr>
<td>DB Size</td>
<td>1151 K</td>
<td>BI Size</td>
<td>2168 K</td>
<td>AI Size</td>
<td>0 K</td>
</tr>
<tr>
<td>FR chain</td>
<td>12 blocks</td>
<td>RM chain</td>
<td>439 blocks</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Shared Memory</td>
<td>966656</td>
<td>Segments</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0 Servers, 10 Users (10 Local, 0 Remote, 10 Batch),0 Apws</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

RETURN - repeat, U - continue uninterrupted, Q - quit:

Figure 20–6: Sample Output for PROMON Activity option

Event

The events that have occurred on the system and the cumulative total number of events and the number of events per second. Table 20–5 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 20–5: Event types

<table>
<thead>
<tr>
<th>Event type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commits</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>Undos</td>
<td>Number of transactions rolled back</td>
</tr>
<tr>
<td>Record Updates</td>
<td>Number of records updated</td>
</tr>
<tr>
<td>Record Reads</td>
<td>Number of records read</td>
</tr>
<tr>
<td>Record Creates</td>
<td>Number of records created</td>
</tr>
<tr>
<td>Record Deletes</td>
<td>Number of records deleted</td>
</tr>
<tr>
<td>DB Writes</td>
<td>Number of database blocks written to disk</td>
</tr>
<tr>
<td>DB Reads</td>
<td>Number of database blocks read from disk</td>
</tr>
<tr>
<td>BI Writes</td>
<td>Number of BI blocks written to disk</td>
</tr>
</tbody>
</table>
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. Table 20–6 defines the process types.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servers</td>
<td>Number of servers running against your database. This is the current value at the time you ran the Activity option, not a cumulative total.</td>
</tr>
<tr>
<td>Users</td>
<td>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.</td>
</tr>
<tr>
<td>Apws</td>
<td>Number of APWs running against your database. This is the current value at the time you ran the Activity option, not a cumulative total.</td>
</tr>
</tbody>
</table>
PROMON Shared Resources option

Displays a variety of system resource usage statistics and startup parameter settings. Broker status information appears only when a broker process is active. Figure 20–7 shows this option’s output. See Chapter 19, “Database Startup Parameters,” for more information on these startup parameters.

![Shared Resources output]

**Figure 20–7:** Sample Output for PROMON Shared Resources option

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 (UNIX only)**

On UNIX systems, 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:

\[
\text{#SEM} = \text{Max-possible-users} (-n) + \text{Max-possible-servers} (-Mn) + 4
\]

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 Chapter 14, “Managing Performance,” 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. Figure 20–8 shows an example of this option’s output.

```
Database version number: 1174
Database state: Open (1)
Database damaged flags: None (0)
Integrity flags: Executing with -r (0)
Database block size (bytes): 1024
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): 03/25/04 16:05
   Most recent database open: 06/04/04 14:19
      Previous database open: 06/04/04 14:19
Local Schema cache time stamp: 03/24/04 20:47
   Most recent .bi file open: 06/04/04 14:20

RETURN - show remaining, U - continue uninterrupted, Q - quit:

   Previous .bi file open: - -
   Time since last truncate bi: 0 (seconds)
   Most recent full backup: 05/26/04 15:06
   Database changed since backup: Yes
   Most recent incremental backup: - -
      Sequence of last incremental: 0

RETURN - repeat, U - continue uninterrupted, Q - quit
```

Figure 20–8: Sample output for PROMON Database Status option
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:

<table>
<thead>
<tr>
<th>Database state</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The database is open</td>
</tr>
<tr>
<td>2</td>
<td>The database is not modified since the last open</td>
</tr>
<tr>
<td>4</td>
<td>The database is being recovered</td>
</tr>
<tr>
<td>8</td>
<td>A database restore is underway</td>
</tr>
<tr>
<td>2048</td>
<td>An index repair is underway</td>
</tr>
</tbody>
</table>

**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. When you choose this option, PROMON displays the menu shown in Figure 20–9.

```
Enter your selection:
    1 Disconnect a User
    2 Unconditional Shutdown
    3 Emergency Shutdown (kill All)
    x Exit
```

Enter choice>

**Figure 20–9: Sample output for PROMON Shut Down Database option**

**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 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:

1. At the **Enter your selection** prompt, enter **R&D**. The R&D main menu appears:

   09/19/08  OpenEdge Release 10 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):

2. 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.

   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
- Backup
- Servers
- Processes/Clients
- Files
- Lock Table
- Buffer Cache
- Logging Summary
- BI Log
- AI Log
- Two-phase Commit
- Startup Parameters
- Shared Resources
- Shared Memory Segments
- AI Extents
- Database Service Manager
- Servers by broker
- Client Database-Request Statement Cache
Database

Displays general database status information. Figure 20–10 shows a sample Database Status display.

<table>
<thead>
<tr>
<th>09/15/06</th>
<th>Status: database</th>
</tr>
</thead>
<tbody>
<tr>
<td>11:01:01</td>
<td></td>
</tr>
</tbody>
</table>

- Database was started at: 09/15/03 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: 09/15/03 9:17
- Previous database open: 09/15/03 9:17
- Local cache file time stamp: 09/10/03 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: 10003

Figure 20–10: PROMON Database Status display

The display in Figure 20–10 contains the following fields:

- **Database state** — The current operating mode of the database. Table 20–7 describes the possible database states.

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open.</td>
<td>The database is open</td>
</tr>
<tr>
<td>Not modified since last open.</td>
<td>The database is closed</td>
</tr>
<tr>
<td>Recovery.</td>
<td>The database is performing crash recovery</td>
</tr>
<tr>
<td>Index repair.</td>
<td>The PROUTIL IDXBUILD utility is running to rebuild indexes</td>
</tr>
<tr>
<td>Restore.</td>
<td>The database is being restored</td>
</tr>
</tbody>
</table>
• **Database damaged flags** — The state of the database. Table 20–8 lists the possible flags.

**Table 20–8: Database damaged flags**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Normal</td>
</tr>
<tr>
<td>Opened with -F</td>
<td>Crash recovery skipped; the database might be damaged</td>
</tr>
<tr>
<td>Crashed with -i</td>
<td>Crashed without integrity; the database is damaged</td>
</tr>
<tr>
<td>Crashed with -r</td>
<td>Crashed with buffered I/O; the database might be damaged</td>
</tr>
</tbody>
</table>

• **Integrity flags** — The integrity mode in which the database is running. Table 20–9 lists the possible flags.

**Table 20–9: Integrity flags**

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>Normal</td>
</tr>
<tr>
<td>Executing with -i</td>
<td>A crash in this state will damage the database</td>
</tr>
<tr>
<td>Executing with -r</td>
<td>A crash in this state might damage the database</td>
</tr>
</tbody>
</table>

• **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 Chapter 19, “Database Startup Parameters,” 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.
Backup

Displays information about database backups. Figure 20–11 shows a sample Backup Status display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Status: backup</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>10/18/06</td>
<td></td>
<td>16:07</td>
</tr>
</tbody>
</table>

Most recent full backup: 10/04/06 11:52
Most recent incremental backup: Never
Database changed since backup: Yes
Sequence of last incremental: 0

Figure 20–11: PROMON Backup Status display

The display in Figure 20–11 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.

Figure 20–12 shows a sample Servers Status display.

<table>
<thead>
<tr>
<th>Sv No</th>
<th>Pid</th>
<th>Type</th>
<th>Protocol</th>
<th>Logins</th>
<th>Pend. Users</th>
<th>Cur. Users</th>
<th>Max. Users</th>
<th>Port Num</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4899</td>
<td>Login</td>
<td>TCP</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>2402</td>
</tr>
<tr>
<td>1</td>
<td>22849</td>
<td>Auto</td>
<td>TCP</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>1025</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>Inactive</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>Inactive</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td></td>
<td>Inactive</td>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 20–10: Server types

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Broker</td>
<td>A broker process. A broker logs in clients and starts remote servers.</td>
</tr>
<tr>
<td>Auto</td>
<td>A server started automatically by the broker.</td>
</tr>
<tr>
<td>Manual</td>
<td>A server started manually (with the Manual Server (-m2) startup parameter) after the broker was started. You might start a server manually for debugging purposes, or on a system where a server cannot be started automatically.</td>
</tr>
<tr>
<td>Login</td>
<td>A secondary login broker. When a database network includes more than one protocol, a secondary login broker performs broker functions for the second protocol.</td>
</tr>
<tr>
<td>Inactive</td>
<td>Placeholder in the server table for servers that have not yet started.</td>
</tr>
</tbody>
</table>

- **Type** — The server type. Table 20–10 describes the possible server types.

- **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 Table 20–11.

Table 20–11: PROMON Processes/Clients Status display columns

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usr</td>
<td>The user number of the process.</td>
</tr>
<tr>
<td>Name</td>
<td>The user name of the process.</td>
</tr>
<tr>
<td>Type</td>
<td>The process type. Table 20–1 lists the possible types.</td>
</tr>
<tr>
<td>Wait</td>
<td>The wait type. Table 20–2 describes the possible types. If no value is displayed, the process is not waiting on any database-related events.</td>
</tr>
<tr>
<td>Login Time</td>
<td>The time when the process logged into the database.</td>
</tr>
<tr>
<td>Start Time</td>
<td>The time when the process started.</td>
</tr>
<tr>
<td>Trans id</td>
<td>The user’s transaction ID.</td>
</tr>
<tr>
<td>Tx Start Time</td>
<td>The time the transaction started.</td>
</tr>
<tr>
<td>Trans State</td>
<td>The state of the transaction. Table 20–4 lists the possible states.</td>
</tr>
<tr>
<td>Pid</td>
<td>The process ID number.</td>
</tr>
<tr>
<td>Serv</td>
<td>The server number serving the remote client.</td>
</tr>
</tbody>
</table>

For certain wait values, an additional field is displayed. Table 20–12 lists the possible values.

Table 20–12: PROMON Processes/Clients Status Wait column additional values

<table>
<thead>
<tr>
<th>Wait type</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>REC</td>
<td>rowid</td>
</tr>
<tr>
<td>SCH</td>
<td>Type of schema lock:</td>
</tr>
<tr>
<td></td>
<td>• 12 — share lock</td>
</tr>
<tr>
<td></td>
<td>• 10 — exclusive lock</td>
</tr>
<tr>
<td>TRAN</td>
<td>taskid</td>
</tr>
<tr>
<td>RGET</td>
<td>rowid</td>
</tr>
<tr>
<td>BKSH</td>
<td>dbkey</td>
</tr>
<tr>
<td>BKEX</td>
<td>dbkey</td>
</tr>
<tr>
<td>BIRD</td>
<td>dbkey (bi)</td>
</tr>
</tbody>
</table>
Processes/Clients

The Processes/Clients menu choices are:

- **All Processes** — Displays status information about OpenEdge processes. Figure 20–13 shows a sample Processes/Clients Status display.

---

**Table 20–12: PROMON Processes/Clients Status Wait column additional values**

<table>
<thead>
<tr>
<th>Wait type</th>
<th>Additional information</th>
</tr>
</thead>
<tbody>
<tr>
<td>BIWR</td>
<td>dbkey (bi)</td>
</tr>
<tr>
<td>AIRD</td>
<td>dbkey (ai)</td>
</tr>
<tr>
<td>ARWR</td>
<td>dbkey (ai)</td>
</tr>
<tr>
<td>TXS</td>
<td>2 — TXE update</td>
</tr>
<tr>
<td></td>
<td>3 — TEX commit</td>
</tr>
<tr>
<td>TXB</td>
<td>1 — TXE share</td>
</tr>
<tr>
<td></td>
<td>2 — TXE update</td>
</tr>
<tr>
<td></td>
<td>3 — TEX commit</td>
</tr>
<tr>
<td>TXE</td>
<td>1 — TXE share</td>
</tr>
<tr>
<td></td>
<td>2 — TXE update</td>
</tr>
<tr>
<td></td>
<td>3 — TEX commit</td>
</tr>
<tr>
<td>TXX</td>
<td>1 — TXE share</td>
</tr>
<tr>
<td></td>
<td>2 — TXE update</td>
</tr>
<tr>
<td></td>
<td>3 — TEX commit</td>
</tr>
</tbody>
</table>

---

**Figure 20–13: PROMON Processes/Clients Status display**

<table>
<thead>
<tr>
<th>09/25/03</th>
<th>All Processes Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:09</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usr</th>
<th>Name</th>
<th>Type</th>
<th>Wait</th>
<th>Trans id</th>
<th>Login time</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SYSTEM</td>
<td>BROK</td>
<td>--</td>
<td>0</td>
<td>09/25/03 08:31</td>
</tr>
<tr>
<td>5</td>
<td>BIW</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>09/25/03 08:33</td>
</tr>
<tr>
<td>6</td>
<td>WDOG</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>09/25/03 08:33</td>
</tr>
<tr>
<td>7</td>
<td>APW</td>
<td>--</td>
<td>0</td>
<td>0</td>
<td>09/25/03 08:34</td>
</tr>
<tr>
<td>8</td>
<td>rgw</td>
<td>MON</td>
<td>--</td>
<td>0</td>
<td>09/25/03 08:57</td>
</tr>
</tbody>
</table>
• **Blocked Clients** — The clients listed are waiting for a database-related resource. Figure 20–14 shows a sample Blocked Clients Status display. The clients listed are waiting for a database-related resource.

09/25/03   Status Blocked Clients
16:10
Usr  Name  Type Wait  Trans id  Login time
2  eadams  SELF  REC  3937  0  09/25/03 14:51

**Figure 20–14: PROMON Blocked Clients Status display**

• **Active Transactions** — Displays status information about active transactions. Figure 20–15 shows a sample Active Transactions Status display.

09/25/03   Active Transactions Status
16:10
Usr  Name  Type Login time  Tx Start Time  Trans id  Trans State
0  rgw  SELF  09/25/03 08:31  2601  Begin

**Figure 20–15: PROMON Active Transactions Status display**

• **Local Interactive Clients** — Displays status information about local interactive client processes. Figure 20–16 shows a sample Local Interactive Clients status display.

09/25/03   Local Interactive Clients Status
16:10
Usr  Name  Type Wait  Trans id  Login time  Pid
0  rgw  BROK  MTX  0  0  09/25/03 10:31  16147

**Figure 20–16: PROMON Local Interactive Clients Status display**

• **Local Batch Clients** — Displays status information about local batch client processes. Figure 20–17 shows a sample Local Batch Clients Status display.

09/25/03   Local Batch Clients Status
16:10
Usr  Name  Type Wait  Trans id  Login time  Pid
0  guy  SELF  MTX  0  402  09/25/03 10:31  678
0  rgw  SELF  REC  3937  0  09/25/03 10:36  687

**Figure 20–17: PROMON Local Batch Clients Status display**
• **Remote Clients** — Displays status information about remote client processes. **Figure 20–18** shows a sample Remote Clients Status display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>User</th>
<th>Name</th>
<th>Type</th>
<th>Wait</th>
<th>Trans</th>
<th>Login Time</th>
<th>Service</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/25/03</td>
<td>16:18</td>
<td>0</td>
<td>rgw</td>
<td>BROK</td>
<td>MTX</td>
<td>0</td>
<td>09/25/03 10:36</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 20–18: PROMON Remote Clients Status display**

• **Background Processes** — Displays status information about background database processes. **Figure 20–19** shows a sample Background Processes Status display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>User</th>
<th>Name</th>
<th>Type</th>
<th>Start Time</th>
<th>Pid</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/25/03</td>
<td>16:10</td>
<td>0</td>
<td>SYSTEM</td>
<td>BROK</td>
<td>09/25/03 10:31</td>
<td>16140</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>rgw</td>
<td>MON</td>
<td>09/25/03 11:12</td>
<td>16145</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>BIW</td>
<td>09/25/03 10:33</td>
<td>16146</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 20–19: PROMON Background Processes Status display**
Files

Displays status information about OpenEdge database files. Figure 20–20 shows a sample Files Status display.

<table>
<thead>
<tr>
<th>File name</th>
<th>Size (KB)</th>
<th>Extend (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/dev/dbs/docsample.db</td>
<td>632</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample.b1</td>
<td>2168</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample.d1</td>
<td>1656</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample_7.d1</td>
<td>248</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample_8.d1</td>
<td>1144</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample_9.d1</td>
<td>248</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample_10.d1</td>
<td>2040</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample_11.d1</td>
<td>2040</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample.a1</td>
<td>504</td>
<td>0</td>
</tr>
<tr>
<td>/dev/dbs/docsample.a2</td>
<td>504</td>
<td>0</td>
</tr>
<tr>
<td>/dev/dbs/docsample.a3</td>
<td>120</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample_1000.d1</td>
<td>632</td>
<td>0</td>
</tr>
<tr>
<td>/dev/dbs/docsample_1000.d2</td>
<td>120</td>
<td>512</td>
</tr>
<tr>
<td>/dev/dbs/docsample_2000.d1</td>
<td>632</td>
<td>0</td>
</tr>
<tr>
<td>/dev/dbs/docsample_2000.d2</td>
<td>120</td>
<td>512</td>
</tr>
</tbody>
</table>

Figure 20–20: PROMON Files Status display

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 Chapter 14, “Managing Performance,” for a description of the lock table and its functions. When you choose this menu option, PROMON gives you three choices:

- Display all lock entries
- Display locks for one user
- Display locks for one RECID

Figure 20–21 shows a sample Lock Table Status display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Status</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/28/06</td>
<td>Lock Table</td>
<td>12:06:54</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usr Name</th>
<th>Trans id</th>
<th>Type</th>
<th>Table</th>
<th>RECID</th>
<th>Flags</th>
<th>Trans State</th>
</tr>
</thead>
<tbody>
<tr>
<td>24 docusr1</td>
<td>174</td>
<td>REC</td>
<td>2</td>
<td>385</td>
<td>X</td>
<td>Begin</td>
</tr>
<tr>
<td>10_docusr2</td>
<td>175</td>
<td>REC</td>
<td>2</td>
<td>385</td>
<td>S</td>
<td>Begin</td>
</tr>
<tr>
<td>23 dba</td>
<td>0</td>
<td>REC</td>
<td>-1</td>
<td>12108</td>
<td>S</td>
<td>Dead</td>
</tr>
</tbody>
</table>

Figure 20–21: PROMON Lock Table Status display

The display contains the following fields:

- **Usr** — The user number of the user who owns the lock entry
- **Name** — The user name
- **Trans id** — The transaction id
- **Type** — The lock type (for example, REC, RGET)
- **Table** — The table number
- **RECID** — The RECID of the record and lock table entry
• **Flags** — Table 20–13 lists the possible flags

Table 20–13: **Lock table flags**

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>Share lock</td>
</tr>
<tr>
<td>X</td>
<td>Exclusive lock</td>
</tr>
<tr>
<td>U</td>
<td>Lock was upgraded from a share lock to an exclusive lock</td>
</tr>
<tr>
<td>L</td>
<td>Limbo—lock is being held until the transaction commits</td>
</tr>
<tr>
<td>Q</td>
<td>Request is queued, and the user is waiting for a conflicting lock held by another user</td>
</tr>
<tr>
<td>P</td>
<td>Purged</td>
</tr>
<tr>
<td>H</td>
<td>Hold</td>
</tr>
</tbody>
</table>

• **Trans State** — Indicates the state of the transaction
Buffer Cache

Displays status information about buffers. Figure 20–22 shows a sample Buffer Cache Status display.

<table>
<thead>
<tr>
<th>09/25/03</th>
<th>Status: Buffer Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:21</td>
<td></td>
</tr>
<tr>
<td>Total buffers:</td>
<td>170</td>
</tr>
<tr>
<td>Hash Table Size:</td>
<td>43</td>
</tr>
<tr>
<td>Used Buffers:</td>
<td>57</td>
</tr>
<tr>
<td>Empty Buffers:</td>
<td>113</td>
</tr>
<tr>
<td>On lru Chain:</td>
<td>168</td>
</tr>
<tr>
<td>On apw queue:</td>
<td>0</td>
</tr>
<tr>
<td>On ckp queue:</td>
<td>0</td>
</tr>
<tr>
<td>Modified buffers:</td>
<td>170</td>
</tr>
<tr>
<td>Marked for ckp:</td>
<td>0</td>
</tr>
<tr>
<td>Last checkpoint number:</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 20–22: PROMON Buffer Cache Status display

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 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.
Logging Summary

Displays status information for the database logging function. Figure 20–23 shows a sample Logging Summary Status display.

<table>
<thead>
<tr>
<th>09/25/06</th>
<th>Status: Logging Summary</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:22</td>
<td></td>
</tr>
</tbody>
</table>

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 journaling: Yes  
After-image I/O: Reliable  
AI Writer status: Not executing

Figure 20–23: PROMON Logging Summary Status display

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 journaling** — 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. Figure 20–24 shows a sample BI Log Status display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Status: BI Log</th>
<th>Time</th>
<th>BI Cluster Age Time</th>
<th>BI Block Size</th>
<th>BI Cluster Size</th>
<th>Number of BI Extents</th>
<th>BI Log Size (kb)</th>
<th>Bytes Free in Current Cluster</th>
<th>Last Checkpoint Was At</th>
<th>Number of BI Buffers</th>
<th>Full Buffers</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/25/03</td>
<td>16:23</td>
<td></td>
<td>60 seconds</td>
<td>8192 bytes</td>
<td>512 kb (524288 bytes)</td>
<td>1</td>
<td>2168</td>
<td>521177 (100%)</td>
<td>09/25/03 13:56</td>
<td>5</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 20–24: PROMON BI Log Status display

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.
**AI Log**

Displays status information for after-image logging, if after-imaging is enabled. Figure 20–25 shows a sample AI Log Status display.

<table>
<thead>
<tr>
<th>09/15/05 Status: AI Log 11:35:24</th>
</tr>
</thead>
<tbody>
<tr>
<td>After-image begin date: 08/18/05 10:47</td>
</tr>
<tr>
<td>After-image new date: 08/18/05 10:47</td>
</tr>
<tr>
<td>After-image open date: 09/15/05 11:31</td>
</tr>
<tr>
<td>After-image generation number: 1</td>
</tr>
<tr>
<td>Number of after-image extents: 3</td>
</tr>
<tr>
<td>Current after-image extent: 13</td>
</tr>
<tr>
<td>Number of AI buffers: 5</td>
</tr>
<tr>
<td>After-image block size: 8192 bytes</td>
</tr>
<tr>
<td>After-image log size: 2168 K</td>
</tr>
</tbody>
</table>

Figure 20–25: PROMON AI Log Status display

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 Chapter 14, “Managing Performance,” for more information.
- **After-image log size** — The size of current busy AI extent, in kilobytes.
Two-phase Commit

Displays status information about two-phase commit, if enabled. Figure 20–26 shows a sample Two-phase Commit Status display.

<table>
<thead>
<tr>
<th>09/25/03</th>
<th>Status: Two-Phase Commit</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:02:19</td>
<td></td>
</tr>
</tbody>
</table>

Coordinator nickname: mydemo
Coordinator priority: 0
After-image journalling: Yes

Figure 20–26: PROMON Two-phase Commit Status display

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 journaling** — Yes, if after-imaging is enabled; no, if it is not.
Startup Parameters

Displays values of OpenEdge database startup parameters. Figure 20–27 shows a sample Startup Parameters Status display.

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum clients</td>
<td>21</td>
</tr>
<tr>
<td>Maximum servers</td>
<td>4</td>
</tr>
<tr>
<td>Maximum clients per server</td>
<td>5</td>
</tr>
<tr>
<td>Lock table size</td>
<td>8192 entries</td>
</tr>
<tr>
<td>Database buffers</td>
<td>4000 (16000 kb)</td>
</tr>
<tr>
<td>Database alternate buffers</td>
<td>2000 (8000 kb)</td>
</tr>
<tr>
<td>Excess shared memory</td>
<td>23 kb</td>
</tr>
<tr>
<td>APW queue check time</td>
<td>100 milliseconds</td>
</tr>
<tr>
<td>APW scan time</td>
<td>1 seconds</td>
</tr>
<tr>
<td>APW buffers to scan</td>
<td>1</td>
</tr>
<tr>
<td>APW max writes/scan</td>
<td>25</td>
</tr>
<tr>
<td>Spinlock tries before timeout</td>
<td>6000</td>
</tr>
<tr>
<td>Before-image buffers</td>
<td>5 (5 kb)</td>
</tr>
<tr>
<td>After-image buffers</td>
<td>5 (5 kb)</td>
</tr>
<tr>
<td>Max number of JTA transactions</td>
<td>100</td>
</tr>
</tbody>
</table>

Figure 20–27: PROMON Startup Parameters Status display

The display contains the following fields:

- **Maximum clients** — The value of the Number of Users (-n) parameter
- **Maximum servers** — The value of the Maximum Servers (-mn) parameter
- **Maximum clients per server** — The value of the Maximum Clients per Server (-Ma) parameter
- **Lock table size** — The value of the Lock Table Entries (-L) parameter
- **Database buffers** — The value of the Blocks in Database Buffers (-B) parameter
- **Database alternate buffers** — The value of the Blocks in Alternate Buffer Poll (-B2) parameter
- **Excess Shared Memory** — The value of the Shared Memory Overflow Size (-Mxs) parameter
- **APW queue check time** — The value of the Page Writer Queue Delay (-pwqdelay) parameter
- **APW scan time** — The value of the Page Writer Scan Delay (-pwsdelay) parameter
- **APW buffers to scan** — The value of the Page Writer Scan (-pwscan) parameter
- **APW max writes/scan** — The value of the Page Writer Maximum Buffers (-pwmax) parameter
- **Spinlock tries before timeout** — The number of times a process will spin (-spin) trying to acquire a resource before waiting
- **Before-image buffers** — The value of the Before-image Buffers (-bibufs) parameter
• **After-image buffers** — The value of the After-image Buffers (-aibufs) parameter

• **Max number of JTA transactions** — The value of the Maximum JTA Transaction Ids (-maxxids) parameter

**Shared Resources**

Displays status information about shared database resources. Figure 20–28 shows a sample Shared Resources Status display.

<table>
<thead>
<tr>
<th>09/25/03</th>
<th>Status: Shared Resources</th>
<th>16:24</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active transactions:</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Lock table entries in use:</td>
<td>8 of 8192</td>
<td></td>
</tr>
<tr>
<td>Lock table high water mark:</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>Number of servers:</td>
<td>0 (5 allocated)</td>
<td></td>
</tr>
<tr>
<td>Total users:</td>
<td>1 (20 allocated)</td>
<td></td>
</tr>
<tr>
<td>Self service:</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Remote:</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Batch:</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Watchdog status:</td>
<td>*** Not executing ***</td>
<td></td>
</tr>
<tr>
<td>BIW status:</td>
<td>Executing</td>
<td></td>
</tr>
<tr>
<td>AIW status:</td>
<td>*** Not executing ***</td>
<td></td>
</tr>
<tr>
<td>Number of page writers:</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Number of monitors:</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Number of semaphores allocated:</td>
<td>29</td>
<td></td>
</tr>
<tr>
<td>Shared memory allocated:</td>
<td>1408 K (1 segments. The last segment was not locked in memory)</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 20–28: PROMON Shared Resources Status display**

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

• **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 Chapter 14, “Managing Performance,” for more information on shared-memory allocation.

### Shared Memory Segments

Displays general information about shared-memory segments allocated to the database. **Figure 20–29** shows a sample Shared Memory Segments Status display.

<table>
<thead>
<tr>
<th>09/20/06</th>
<th>Status: Shared Memory Segments</th>
</tr>
</thead>
<tbody>
<tr>
<td>13:17:44</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Seg</th>
<th>Id</th>
<th>Size</th>
<th>Used</th>
<th>Free</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>91242</td>
<td>858972160</td>
<td>849424036</td>
<td>9548124</td>
</tr>
</tbody>
</table>

**Figure 20–29: PROMON Shared Memory Segments Status display**

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. Figure 20–30 shows a sample AI Extents status display.

<table>
<thead>
<tr>
<th>Area</th>
<th>Status</th>
<th>Type</th>
<th>File Number</th>
<th>Size (KBytes)</th>
<th>Extent Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>BUSY</td>
<td>Fix</td>
<td>1</td>
<td>505</td>
<td>/usr1/101A/doc_sample.a1</td>
</tr>
<tr>
<td>14</td>
<td>EMPTY</td>
<td>Fix</td>
<td>0</td>
<td>505</td>
<td>/usr1/101A/doc_sample.a2</td>
</tr>
<tr>
<td>15</td>
<td>EMPTY</td>
<td>Var</td>
<td>0</td>
<td>121</td>
<td>/usr1/101A/doc_sample.a3</td>
</tr>
</tbody>
</table>

**Figure 20–30: PROMON AI Extents status display**

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.

<table>
<thead>
<tr>
<th>Date</th>
<th>Status: Database Service Manager</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/29/06</td>
<td></td>
</tr>
<tr>
<td>10:51:13</td>
<td></td>
</tr>
</tbody>
</table>

Communication Area Size : 7.00 KB
Total Message Entries : 64
Free Message Entries : 63
Used Message Entries : 1

Registered Database Service Objects

<table>
<thead>
<tr>
<th>Name</th>
<th>Rdy</th>
<th>Status</th>
<th>Messages Locked by</th>
</tr>
</thead>
<tbody>
<tr>
<td>OpenEdge Replication Server</td>
<td>Y</td>
<td>BUSYNP</td>
<td>1</td>
</tr>
<tr>
<td>OpenEdge RDBMS</td>
<td>Y</td>
<td>REG</td>
<td>0</td>
</tr>
</tbody>
</table>

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
- **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
Table 20–14 lists the possible values for Status.

Table 20–14: Status values

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RUN</td>
<td>Running</td>
</tr>
<tr>
<td>ACT</td>
<td>Active</td>
</tr>
<tr>
<td>INACT</td>
<td>Inactive—the service is registered, but not active</td>
</tr>
<tr>
<td>BUSY</td>
<td>Busy</td>
</tr>
<tr>
<td>REG</td>
<td>Registered</td>
</tr>
<tr>
<td>PREREG</td>
<td>Pre-registered. The service is known, but it not ready for use</td>
</tr>
<tr>
<td>UNK</td>
<td>Unknown</td>
</tr>
</tbody>
</table>

- **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.

<table>
<thead>
<tr>
<th>Sv No</th>
<th>Pid</th>
<th>Type</th>
<th>Protocol</th>
<th>Logins</th>
<th>Pend. Users</th>
<th>Cur. Users</th>
<th>Max. Users</th>
<th>Port Num</th>
</tr>
</thead>
</table>

The display contains the following fields:

- **Service number** — Indicates the service number
- **Pid** — The process ID number
- **Type** — The process type—Table 20–1 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.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>12/10/07</td>
<td>09:06:10</td>
<td>OpenEdge Release 10 Monitor (R&amp;D)</td>
</tr>
</tbody>
</table>

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

The display contains the following options:

- **Activate For Selected Users** — Activates Client Database-Request Statement Caching for one or more clients. For more information, see the “Activating database-request statement caching for selected users” section on page 20–50.

- **Activate For All Users** — Activates Client Database-Request Statement Caching for all clients. For more information, see the “Activating database-request statement caching for all users” section on page 20–51.

- **Activate For All Future Users** — Activates Client Database-Request Statement Caching for all future client connections to the database. For more information, see the “Activating database-request statement caching for all future users” section on page 20–52.

- **Deactivate For Selected Users** — Deactivates Client Database-Request Statement Caching for one or more clients. For more information, see the “Deactivating database-request statement caching” section on page 20–53.

- **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 the “Viewing database-request statement caching” section on page 20–54.

- **Specify Directory for Statement Cache Files** — Specify the directory for all temporary database-request statement cache files. For more information, see the “Specifying a directory for database-request statement caching files” section on page 20–56.
Activating database-request statement caching for selected users

If you choose option 1 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):

This display includes the following fields:

- **Single** — Only the current ABL program and line number, or a single SQL statement is reported by the ABL client.
- **Stack** — The current ABL program and line number. This option displays up to 31 prior ABL program names and line numbers, or a single SQL statement.
- **One Time Request** — The ABL or SQL client report tracing information once. Once reported, database-request statement caching is turned off.
- **Quit** — Quit this function and return to the Client Database-Request Statement Caching menu.

If you choose option 2 to activate the Stack type of 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): 2

12/18/07 OpenEdge Release 10 Monitor (R&D)
12:23:14 Activate For Selected Users

<table>
<thead>
<tr>
<th>User</th>
<th>Name</th>
<th>Type</th>
<th>Login time</th>
<th>Serv IPV#</th>
<th>Remote Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>User-5</td>
<td>SELF</td>
<td>12/18/07 12:23</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User-6</td>
<td>SELF</td>
<td>12/18/07 12:24</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>User-7</td>
<td>SELF</td>
<td>12/18/07 12:24</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>User-8</td>
<td>SELF</td>
<td>12/18/07 12:24</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Enter user number, P, T, or X (? for help):

If you select Single, Stack, or One Time Request for the type of statement caching to activate, the list of database connections displays:

- All of the self-serve connections at the top of the list
- Remote database servers, followed by the remote connections they are servicing
- Each user group, ordered by user number
Once the user information is displayed, you can specify a user number. Based on the user number, the following actions are performed:

- Database-request statement caching is activated if the user number is valid, or if the user is specified in the list. 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 cancelled. You must deactivate for selected users, all users, or all future users.

- 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.

- If you enter a valid numeric string, you can continue entering valid user numbers.

**Activating database-request statement caching for all users**

If you choose option 2 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): 2
```

This display includes the following fields:

- **Single** — Only the current ABL program and line number, or a single SQL statement is reported by the ABL client.

- **Stack** — The current ABL program and line number. This option displays up to 31 prior ABL program names and line numbers, or a single SQL statement.

- **One Time Request** — The ABL or SQL client report tracing information once. Once reported, database-request statement caching is turned off.

- **Quit** — Quit this function and return to the Client Database-Request Statement Caching menu.

If you choose option 2 to activate the **Stack** type of statement caching for all clients, the following menu appears:

```
Enter the type of statement caching to activate (1-Single, 2-Stack, Q-Quit): 2
```

12/18/07 Status: Activate For All Users
12:53:58

Database-Request Statement Caching was Activated for 2 users.

Enter <return>, R, P, T, or X (? for help):
Once the type of database-request statement caching is selected, tracing is activated for all clients currently connected to the database. An informational message indicates if database-request statement caching was successful, or if caching failed.

**Note:** Once activated, database-request statement caching remains activated until deactivated.

### Activating database-request statement caching for all future users

If you choose **option 3** 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): 
```

This display includes the following fields:

- **Single** — Only the current ABL program and line number, or a single SQL statement is reported by the ABL client.
- **Stack** — The current ABL program and line number. This option displays up to 31 prior ABL program names and line numbers, or a single SQL statement.
- **Quit** — Quit this function and return to the Client Database-Request Statement Caching menu.

If you choose **option 2** to activate the **Stack** type of statement caching for all future clients, the following menu appears:

```
Enter the type of statement caching to activate (1-Single, 2-Stack, Q-Quit): 2 
```

```
12/18/07        Status: Activate Client Tracing For All Future Users
12:58:12
Database-request statement caching was Activated for all future database connections.
```

Enter <return>, R, P, T, or X (? for help):

Once the type of database-request statement caching is selected, tracing is activated for all future users. An informational message indicates if database-request statement caching was successful, or if caching failed.

**Note:** Once activated, database-request statement caching remains activated until deactivated.
Deactivating database-request statement caching

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 the “Viewing database-request statement caching” section on page 20–54.

If you choose option 4 to deactivate database-request statement caching for selected users, the following menu appears:

<table>
<thead>
<tr>
<th>User</th>
<th>Name</th>
<th>Type</th>
<th>Login time</th>
<th>Serv IPV#</th>
<th>Remote Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>User-5</td>
<td>SELF</td>
<td>12/18/07 13:33</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User-6</td>
<td>SELF</td>
<td>12/18/07 13:34</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>User-7</td>
<td>SELF</td>
<td>12/18/07 13:34</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>User-8</td>
<td>SELF</td>
<td>12/18/07 13:34</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Enter user number, P, T, or X (? for help):

In the **Deactivate For Selected Users** window the list of database connections displays:

- Users that have database-request statement caching activated
- All of the self-serve connections at the top of the list
- Remote database servers, followed by the remote connections they are servicing
- Each user group, ordered by user number

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:

- Database-request statement caching is deactivated if the user number is valid, and if the user is specified in the list. 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.

- 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.

- If you enter a valid numeric string, you can continue entering valid user numbers.
Viewing database-request statement caching

If you choose option 7 to view the database-request statement cache, the following menu appears:

<table>
<thead>
<tr>
<th>User number</th>
<th>User Name</th>
<th>Type</th>
<th>Login time</th>
<th>Serv Type</th>
<th>Cache Update</th>
<th>IPV#</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>User-5</td>
<td>SELF</td>
<td>12/18/07 13:33</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>User-6</td>
<td>SELF</td>
<td>12/18/07 13:34</td>
<td>L1</td>
<td>12/18/07 13:35</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>User-7</td>
<td>SELF</td>
<td>12/18/07 13:34</td>
<td>L1</td>
<td>12/18/07 13:35</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>User-8</td>
<td>SELF</td>
<td>12/18/07 13:34</td>
<td>L1</td>
<td>12/18/07 13:35</td>
<td></td>
</tr>
</tbody>
</table>

Enter user number, P, T, or X (? for help): 6

<table>
<thead>
<tr>
<th>User number</th>
<th>User Name</th>
<th>User type</th>
<th>Login date/time</th>
<th>Statement caching type</th>
<th>Statement caching last updated</th>
<th>Statement cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>User-6</td>
<td>SELF</td>
<td>12/18/07 13:34</td>
<td>SQL Statement or Single ABL Program Name</td>
<td>12/18/07 13:35</td>
<td>102:/bin/read_all_records_again.p_01234_.p</td>
</tr>
</tbody>
</table>

Enter user number, P, T, or X (? for help):
Table 20–15 provides information about fields in the View Client Statement Cache display.

**Table 20–15: View client statement cache display fields**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Number</td>
<td>The user number associated with the database connection.</td>
</tr>
<tr>
<td>User Name</td>
<td>The user name associated with the connected user.</td>
</tr>
<tr>
<td>User Type</td>
<td>The type of database connection.</td>
</tr>
<tr>
<td>Login date/time</td>
<td>The date and time the user connected to the database.</td>
</tr>
<tr>
<td>Statement cache type</td>
<td><strong>SQL statement or Single ABL Program Name</strong> — The statement cache information contains a single SQL statement or a partial ABL program line number.</td>
</tr>
<tr>
<td></td>
<td><strong>SQL statement or Partial ABL Program Name</strong> — The statement cache information contains a single SQL statement or a partial ABL program stack containing at most 32 ABL program names and line numbers.</td>
</tr>
<tr>
<td></td>
<td><strong>One Time SQL Statement or Partial ABL Program Name</strong> — The statement cache information contains a one time request of a single SQL statement or a partial ABL program stack containing at most 32 ABL program names and line numbers.</td>
</tr>
<tr>
<td>Statement cache last updated</td>
<td>The date and time the database-request statement cache was updated.</td>
</tr>
<tr>
<td>Statement cache</td>
<td>The SQL statement or ABL program information for the last database request made by the client. This field may change, if:</td>
</tr>
<tr>
<td></td>
<td>• 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.</td>
</tr>
<tr>
<td></td>
<td>• The client is a SQL client; the SQL statement is displayed.</td>
</tr>
</tbody>
</table>
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:

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.
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
- Servers
- Buffer Cache
- Page Writers
- BI Log
- AI Log
- Lock Table
- I/O Operations By Type
- I/O Operations by File
- Space Allocation
- Index
- Record
- Other

The pages of activity displays end with this prompt:

Table 20–16 describes the actions PROMON takes, based on the input received.

Table 20–16: Activity display prompt

<table>
<thead>
<tr>
<th>If you type . . .</th>
<th>PROMON will . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>ENTER</td>
<td>Display the next page of the current display if one exists, otherwise return to the previous menu.</td>
</tr>
<tr>
<td>A</td>
<td>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.</td>
</tr>
<tr>
<td>L</td>
<td>Load activity counters from start of session. Use this option to reset the activity counters after a sampling a smaller time slice.</td>
</tr>
<tr>
<td>R</td>
<td>Repeat the current display.</td>
</tr>
<tr>
<td>S</td>
<td>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.</td>
</tr>
<tr>
<td>U</td>
<td>Update activity counters. The current display then restarts. The data shown include changes which occurred since the initial set of data was collected.</td>
</tr>
<tr>
<td>Z</td>
<td>Zero the activity counters so updates show changes from this time forward.</td>
</tr>
<tr>
<td>P</td>
<td>Return to the previous menu.</td>
</tr>
<tr>
<td>T</td>
<td>Return to the top-level (main) menu.</td>
</tr>
<tr>
<td>X</td>
<td>Exit from the OpenEdge Monitor utility.</td>
</tr>
<tr>
<td>?</td>
<td>Display help about the entry choices.</td>
</tr>
</tbody>
</table>
Summary

Displays general information about database activity. Figure 20–31 shows a sample Summary Activity display.

<table>
<thead>
<tr>
<th>Event</th>
<th>Total</th>
<th>Per Sec.</th>
<th>Event</th>
<th>Total</th>
<th>Per Sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commits</td>
<td>1208</td>
<td>0.7</td>
<td>DB Reads</td>
<td>57</td>
<td>0.0</td>
</tr>
<tr>
<td>Undos</td>
<td>0</td>
<td>0.0</td>
<td>DB Writes</td>
<td>3</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Reads</td>
<td>1334</td>
<td>0.8</td>
<td>BI Reads</td>
<td>13</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Updates</td>
<td>1208</td>
<td>0.7</td>
<td>BI Writes</td>
<td>131</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Creates</td>
<td>0</td>
<td>0.0</td>
<td>AI Writes</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Deletes</td>
<td>0</td>
<td>0.0</td>
<td>Checkpoints</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Locks</td>
<td>3625</td>
<td>2.3</td>
<td>Flushed at chkpt</td>
<td>0</td>
<td>0.0</td>
</tr>
<tr>
<td>Record Waits</td>
<td>0</td>
<td>0.0</td>
<td>Active trans:</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

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

Figure 20–31: PROMON Summary Activity display

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
Summary

- **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. Table 20–17 defines the process types.

**Table 20–17: Process types**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Servers</td>
<td>Number of servers running against your database. This is the current value at the time you run the Activity option, not a cumulative total.</td>
</tr>
<tr>
<td>Clients</td>
<td>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.</td>
</tr>
<tr>
<td>APWs</td>
<td>Number of APWs running against your database. This is the current value at the time you run the Activity option, not a cumulative total.</td>
</tr>
</tbody>
</table>
Servers

Displays information about server activity. Figure 20–32 shows a sample Servers Activity display. A separate page is displayed for each server.

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity: Servers</th>
<th>from 09/25/03 13:56 to 09/25/03 13:57 (19 sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/25/03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16:28</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Messages received</td>
<td>204</td>
<td>131</td>
</tr>
<tr>
<td>Messages sent</td>
<td>152</td>
<td>98</td>
</tr>
<tr>
<td>Bytes received</td>
<td>13252</td>
<td>8549</td>
</tr>
<tr>
<td>Bytes sent</td>
<td>40683</td>
<td>26247</td>
</tr>
<tr>
<td>Records received</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Records sent</td>
<td>122</td>
<td>79</td>
</tr>
<tr>
<td>Queries received</td>
<td>54</td>
<td>35</td>
</tr>
<tr>
<td>Time slices</td>
<td>51</td>
<td>32</td>
</tr>
<tr>
<td>Total Per Min Per Sec Per Tx</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Activity for Server 1

Figure 20–32: PROMON Servers Activity display

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. Figure 20–33 shows a sample Buffer Cache Activity display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity: Buffer Cache</th>
<th>Time Period: 06/30/09 11:23 to 06/30/09 11:33 (9 min 23 sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Buffer Pool</td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Logical reads</td>
<td>28841</td>
<td>3074</td>
</tr>
<tr>
<td>Logical writes</td>
<td>32703</td>
<td>3485</td>
</tr>
<tr>
<td>O/S reads</td>
<td>358</td>
<td>38</td>
</tr>
<tr>
<td>O/S writes</td>
<td>748</td>
<td>80</td>
</tr>
<tr>
<td>Checkpoints</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>Marked to checkpoint</td>
<td>525</td>
<td>56</td>
</tr>
<tr>
<td>Flushed at checkpoint</td>
<td>463</td>
<td>49</td>
</tr>
<tr>
<td>Writes deferred</td>
<td>32161</td>
<td>3427</td>
</tr>
<tr>
<td>LRU skips</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>LRU writes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>APW enqueues</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Database buffer pool hit ratio: 99 %</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Primary Buffer Pool                     | Total | Per Min | Per Sec | Per Tx     |
| Logical reads                           | 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                    | Total | Per Min | Per Sec | Per Tx     |
| 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 replacement policy disabled.        |       |         |         |            |

Figure 20–33: PROMON Buffer Cache Activity display
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. Figure 20–34 shows a sample Page Writers Activity display.

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Activity: Page Writers</th>
<th>Total DB writes</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/26/03 16:29</td>
<td>from 09/25/03 13:56 to 09/26/03 11:23 (21 hrs 27 min)</td>
<td>3</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Total DB writes</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Scan writes</td>
<td>8</td>
<td>25</td>
<td>0.42</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>APW queue writes</td>
<td>2</td>
<td>6</td>
<td>0.10</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Ckp queue writes</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Scan cycles</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Buffers scanned</td>
<td>173</td>
<td>0</td>
<td>0.11</td>
<td>0.14</td>
</tr>
<tr>
<td></td>
<td>Checkpoints</td>
<td>8211</td>
<td>0</td>
<td>5.22</td>
<td>6.79</td>
</tr>
<tr>
<td></td>
<td>Marked at checkpoint</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>Flushed at checkpoint</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Number of APWs:</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 20–34: PROMON Page Writers Activity display**

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. Figure 20–35 shows a sample BI Log Activity display.

![Figure 20–35: PROMON BI Log Activity display](image)

<table>
<thead>
<tr>
<th>Activity</th>
<th>BI Log</th>
<th>09/28/06 10:50 to 09/28/06 15:22 (4 hrs 32 min)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Total BI writes</td>
<td>1517</td>
<td>6</td>
</tr>
<tr>
<td>BIW BI writes</td>
<td>956</td>
<td>4</td>
</tr>
<tr>
<td>Records written</td>
<td>116270</td>
<td>428</td>
</tr>
<tr>
<td>Bytes written</td>
<td>9996664</td>
<td>36768</td>
</tr>
<tr>
<td>Total BI Reads</td>
<td>386</td>
<td>1</td>
</tr>
<tr>
<td>Records read</td>
<td>26923</td>
<td>99</td>
</tr>
<tr>
<td>Bytes read</td>
<td>2330572</td>
<td>8572</td>
</tr>
<tr>
<td>Clusters closed</td>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>Busy buffer waits</td>
<td>588</td>
<td>2</td>
</tr>
<tr>
<td>Empty buffer waits</td>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>Log force waits</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Log force writes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Partial writes</td>
<td>222</td>
<td>1</td>
</tr>
<tr>
<td>Input buffer hits</td>
<td>388</td>
<td>1</td>
</tr>
<tr>
<td>Output buffer hits</td>
<td>135</td>
<td>0</td>
</tr>
<tr>
<td>Mod buffer hits</td>
<td>431</td>
<td>2</td>
</tr>
<tr>
<td>BO buffer hits</td>
<td>53169</td>
<td>196</td>
</tr>
</tbody>
</table>

Figure 20–35: PROMON BI Log Activity display

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. Figure 20–36 shows a sample AI Log Activity display.

<table>
<thead>
<tr>
<th>09/13/03 11:36:56</th>
<th>Activity: AI Log from 09/12/03 13:56 to 09/13/03 11:23 (21 hrs 27 min)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total AI writes</td>
<td>131  0  0.08  0.10</td>
</tr>
<tr>
<td>AIW AI writes</td>
<td>127  0  0.08  0.10</td>
</tr>
<tr>
<td>Records written</td>
<td>3630 25 2.31 3.00</td>
</tr>
<tr>
<td>Bytes written</td>
<td>129487 0 82.42 107.19</td>
</tr>
<tr>
<td>Busy buffer waits</td>
<td>13  0  0.00  0.01</td>
</tr>
<tr>
<td>Buffer not avail</td>
<td>0  0  0.00  0.00</td>
</tr>
<tr>
<td>Partial Writes</td>
<td>4  0  0.00  0.00</td>
</tr>
<tr>
<td>Log force waits</td>
<td>0  0  0.00  0.00</td>
</tr>
</tbody>
</table>

Figure 20–36: PROMON AI Log Activity display

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.

Figure 20–37 shows a sample Lock Table Activity display.

<table>
<thead>
<tr>
<th>04/29/04</th>
<th>Activity: Lock Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>15:11:21</td>
<td>04/29/04 15:05 to 04/29/04 15:10 (5 min 14 sec)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Requests</th>
<th>Total</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share</td>
<td>3715</td>
<td>710</td>
<td>11.83</td>
<td>61.92</td>
</tr>
<tr>
<td>Intent Share</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Exclusive</td>
<td>8</td>
<td>2</td>
<td>0.03</td>
<td>0.13</td>
</tr>
<tr>
<td>Intent Exclusive</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Share Intent Excl</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Upgrade</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Record Get Lock</td>
<td>3505</td>
<td>670</td>
<td>11.16</td>
<td>58.42</td>
</tr>
<tr>
<td>Table Lock</td>
<td>20</td>
<td>4</td>
<td>0.06</td>
<td>0.33</td>
</tr>
<tr>
<td>Record Lock</td>
<td>198</td>
<td>38</td>
<td>0.63</td>
<td>3.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Grants</th>
<th>Total</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share</td>
<td>3715</td>
<td>710</td>
<td>11.83</td>
<td>61.92</td>
</tr>
<tr>
<td>Intent Share</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Exclusive</td>
<td>8</td>
<td>2</td>
<td>0.03</td>
<td>0.13</td>
</tr>
<tr>
<td>Intent Exclusive</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Share Intent Excl</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Upgrade</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Record Get Lock</td>
<td>3505</td>
<td>670</td>
<td>11.16</td>
<td>58.42</td>
</tr>
<tr>
<td>Table Lock</td>
<td>20</td>
<td>4</td>
<td>0.06</td>
<td>0.33</td>
</tr>
<tr>
<td>Record Lock</td>
<td>198</td>
<td>38</td>
<td>0.63</td>
<td>3.30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Waits</th>
<th>Total</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Share</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Intent Share</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Exclusive</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Intent Exclusive</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Share Intent Excl</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Upgrade</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Record Get Lock</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Table Lock</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Record Lock</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Requests Cancelled</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Downgrades</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Shared Find</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Exclusive Find</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 20–37: PROMON Lock Table Activity display

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. Figure 20–38 shows a sample I/O Operations by Type Activity display.

### Table: I/O Operations By Type

<table>
<thead>
<tr>
<th>Date/Time</th>
<th>Activity: I/O Operations by Type</th>
<th>From</th>
<th>To</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/25/03 16:31</td>
<td>09/25/03 15:31 to 09/25/03 16:04</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type</th>
<th>Total</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database reads</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index blocks</td>
<td>13</td>
<td>0</td>
<td>0.00</td>
<td>0.01</td>
</tr>
<tr>
<td>Data blocks</td>
<td>44</td>
<td>0</td>
<td>0.02</td>
<td>0.03</td>
</tr>
<tr>
<td>BI reads</td>
<td>13</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>AI Reads</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.05</td>
</tr>
<tr>
<td>Total reads</td>
<td>70</td>
<td>0</td>
<td>0.04</td>
<td>0.05</td>
</tr>
<tr>
<td>Database writes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index blocks</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Data blocks</td>
<td>3</td>
<td>0</td>
<td>0.08</td>
<td>0.00</td>
</tr>
<tr>
<td>BI writes</td>
<td>131</td>
<td>0</td>
<td>0.00</td>
<td>0.10</td>
</tr>
<tr>
<td>AI writes</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Total writes</td>
<td>134</td>
<td>0</td>
<td>0.08</td>
<td>0.11</td>
</tr>
</tbody>
</table>

**Figure 20–38:** PROMON I/O Operations by Type Activity display

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. Figure 20–39 shows a sample I/O Operations by File Activity display.

<table>
<thead>
<tr>
<th>File Path</th>
<th>Activity: I/O Operations by File</th>
<th>10/18/06 12:14 to 10/18/06 12:15 (11 sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>/scratch/shannon/101B/sep25.db</td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Reads</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>Writes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Extends</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>/scratch/shannon/101B/sep25.b1</td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Reads</td>
<td>15</td>
<td>82</td>
</tr>
<tr>
<td>Writes</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Extends</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>/scratch/shannon/101B/sep25.d1</td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Reads</td>
<td>25</td>
<td>136</td>
</tr>
<tr>
<td>Writes</td>
<td>3</td>
<td>16</td>
</tr>
<tr>
<td>Extends</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>/scratch/shannon/101B/sep25_7.d1</td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Reads</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>Writes</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Extends</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Enter <return> for more, A, L, R, S, U, Z, P, T, or X (? for help):

Figure 20–39: PROMON I/O Operations by File Activity display

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. Figure 20–40 shows a sample Space Allocation Activity display.

Figure 20–40: PROMON Space Allocation Activity display

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. Figure 20–41 shows a sample Index Activity display.

<table>
<thead>
<tr>
<th>09/14/03</th>
<th>Activity: Index</th>
</tr>
</thead>
<tbody>
<tr>
<td>17:09:35</td>
<td>from 09/14/03 17:01 to 09/14/03 17:07 (6 min)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Operation</th>
<th>Total</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Find index entry</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Create index entry</td>
<td>312</td>
<td>1</td>
<td>0.02</td>
<td>104.00</td>
</tr>
<tr>
<td>Delete index entry</td>
<td>8</td>
<td>0</td>
<td>0.00</td>
<td>2.67</td>
</tr>
<tr>
<td>Remove locked entry</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Split block</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Free block</td>
<td>0</td>
<td>0</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>

Figure 20–41: PROMON Index Activity display

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. Figure 20–42 shows a sample Record Activity display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity: Record</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>09/14/03</td>
<td>From 09/14/03 9:24 to 09/14/03 14:38 (5 hrs 14 min)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Total</th>
<th>Per Min</th>
<th>Per Sec</th>
<th>Per Tx</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read record</td>
<td>2374</td>
<td>0</td>
<td>0.13</td>
</tr>
<tr>
<td>Update record</td>
<td>18</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Create record</td>
<td>10</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Delete record</td>
<td>3</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Fragments read</td>
<td>2610</td>
<td>8</td>
<td>0.14</td>
</tr>
<tr>
<td>Fragments created</td>
<td>24</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Fragments deleted</td>
<td>17</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Fragments updated</td>
<td>4</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td>Bytes read</td>
<td>409795</td>
<td>1307</td>
<td>21.7</td>
</tr>
<tr>
<td>Bytes created</td>
<td>1813</td>
<td>6</td>
<td>0.10</td>
</tr>
<tr>
<td>Bytes deleted</td>
<td>1310</td>
<td>4</td>
<td>0.07</td>
</tr>
<tr>
<td>Bytes updated</td>
<td>1042</td>
<td>3</td>
<td>0.06</td>
</tr>
</tbody>
</table>

Figure 20–42: PROMON Record Activity display

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. Figure 20–43 shows a sample Other Activity display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity: Other</th>
<th>From 09/14/03 17:01 to 09/14/03 17:01 (10 sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Commit</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>Undo</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Wait on semaphore</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Flush master block</td>
<td>3</td>
<td>18</td>
</tr>
</tbody>
</table>

Figure 20–43: PROMON Other Activity display

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
- **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. The menu includes the following:

- Performance Indicators
- I/O Operations by Process
- Lock Requests by User
- Checkpoints
- I/O Operations by User by Table
- I/O Operations by User by Index

Performance Indicators

Displays activity statistics related to performance. Figure 20–44 shows a sample Performance Indicators display.

<table>
<thead>
<tr>
<th>Date</th>
<th>Activity: Performance Indicators</th>
<th>Time Range: 09/15/03 10:01 to 09/19/03 13:26 (99 hrs 24 min)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Total</td>
<td>Per Min</td>
</tr>
<tr>
<td>Commits</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Undos</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Index Operations</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Record Operations</td>
<td>3</td>
<td>18</td>
</tr>
<tr>
<td>Total o/s i/o</td>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>Total o/s reads</td>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>Total o/s writes</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Background o/s writes</td>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>Partial log writes</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>Database extends</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Total waits</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Lock waits</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Resource waits</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Latch timeouts</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

buffer pool hit rate: 68%

Figure 20–44: PROMON Performance Indicators display

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 Chapter 14, “Managing Performance,” 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. Figure 20–45 shows a sample I/O Operations by Process display.

<table>
<thead>
<tr>
<th>Usr</th>
<th>Name</th>
<th>Database Access</th>
<th>Database Read</th>
<th>Database Write</th>
<th>BI Read</th>
<th>BI Write</th>
<th>AI Read</th>
<th>AI Write</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SYSTEM</td>
<td>22</td>
<td>8</td>
<td>3</td>
<td>13</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>rgw</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>127</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>3955</td>
<td>49</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 20–45: PROMON I/O Operations by Process display

This display shows the following information:

- **Usr** — The user number of the process
- **Name** — The user name of the process
- **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. Figure 20–46 shows a sample Lock Requests by User display.

<table>
<thead>
<tr>
<th>Usr Num</th>
<th>User Name</th>
<th>--- Record Locks ---</th>
<th>--- Record Waits ---</th>
<th>--- Trans Locks ---</th>
<th>--- Trans Waits ---</th>
<th>--- Schema Locks ---</th>
<th>--- Schema Waits ---</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>SYSTEM</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>rgw</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>3625</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 20–46: PROMON Lock Requests by User display

This display shows the following information:

- **Usr Num** — The user number
- **User Name** — The user name
- **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. Figure 20–47 shows a sample Checkpoints display.

<table>
<thead>
<tr>
<th>Ckpt No.</th>
<th>Time</th>
<th>Len</th>
<th>Freq</th>
<th>Dirty</th>
<th>CPT Q</th>
<th>Scan</th>
<th>APW Q</th>
<th>Flushes</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>14:50:51</td>
<td>6</td>
<td>0</td>
<td>62</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>14:50:48</td>
<td>0</td>
<td>3</td>
<td>61</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>23</td>
<td>14:50:43</td>
<td>0</td>
<td>5</td>
<td>61</td>
<td>61</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>14:50:36</td>
<td>6</td>
<td>7</td>
<td>60</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>14:50:25</td>
<td>0</td>
<td>11</td>
<td>71</td>
<td>70</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td>14:50:05</td>
<td>17</td>
<td>20</td>
<td>68</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>14:49:51</td>
<td>0</td>
<td>14</td>
<td>68</td>
<td>65</td>
<td>0</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>18</td>
<td>14:48:20</td>
<td>91</td>
<td>91</td>
<td>76</td>
<td>63</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 20–47: PROMON Checkpoints display

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
- **Freq** — The length of time between checkpoint starts
- **Dirty** — The number of modified blocks scheduled to be written
- **Database Writes CPT Q** — The number of blocks written from the checkpoint queue by the APWs
- **Database Writes Scan** — The number of blocks written by the APWs during the scan cycle
- **Database Writes APW Q** — The number of blocks written by the APW queue and replaced on the least recently used (LRU) chain by APWs
- **Database Writes Flushes** — The total number of blocks not written during the checkpoint that had to be written all at once at the end of the checkpoint
I/O Operations by User by Table

Displays I/O operations by user and by table. Figure 20–48 shows a sample I/O Operations by User by Table display.

<table>
<thead>
<tr>
<th>User</th>
<th>Table</th>
<th>Reads</th>
<th>Updates</th>
<th>Creates</th>
<th>Deletes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Enter <return> for more, A, L, R, S, U, Z, P, T, or X (? for help):

Figure 20–48: PROMON I/O Operations by User by Table display

This display shows the following information:

- **User** — The user number
- **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 Chapter 19, “Database Startup Parameters,” for more information on these parameters.
I/O Operations by User by Index

Displays information regarding I/O operations by user and by index. Figure 20–49 shows a sample I/O Operations by User by Index display.

<table>
<thead>
<tr>
<th>User</th>
<th>Index</th>
<th>Reads</th>
<th>Creates</th>
<th>Deletes</th>
<th>Splits</th>
<th>Block Dels</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>17</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>22</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>23</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>24</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>26</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>27</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>28</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>29</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>31</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>32</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>33</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Enter <return> for more, A, L, R, S, U, Z, P, T, or X (? for help):

Figure 20–49: PROMON I/O Operations by User by Index display
This display shows the following information:

- **User** — The user number
- **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 number 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 Chapter 19, “Database Startup Parameters,” for more information on these parameters.
R&D Administrative Functions

The **R&D Administrative Functions** menu provides options that let you view and change settings, including:

- Check Active Transaction Status
- Check Two-Phase Transactions
- Resolve Limbo Transactions
- Adjust Latch Options
- Adjust Page Writer Options
- Restricted Options
- Terminate a Server
- Enable/disable block level consistency check

**Check Active Transaction Status**

Displays information about active transactions. Figure 20–50 shows a sample display.

<table>
<thead>
<tr>
<th>09/25/03</th>
<th>OpenEdge Release 10 Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:19:34</td>
<td>Check Active Transaction Status</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Usr</th>
<th>Name</th>
<th>Txt start time</th>
<th>Trans id</th>
<th>Trans State</th>
<th>Crd Nam</th>
<th>Crd Tx Id</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>rjw</td>
<td>09/25/03 16:12</td>
<td>178</td>
<td>Active</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

**Figure 20–50: PROMON Check Active Transaction Status display**

The display shows the following information:

- **Usr** — The user number of the process
- **Name** — The user name of the process
- **Tx start time** — The transaction’s start time
- **Trans id** — The transaction’s ID
- **Trans State** — The transactions state. Table 20–4 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. When you choose this option, PROMON displays the menu shown in Figure 20–51.

| 1 Abort a Limbo Transaction |
| 2 Commit a Limbo Transaction |
| Q Quit |

Enter choice>

Figure 20–51: PROMON Resolve Limbo Transactions menu

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. Figure 20–52 shows a sample Adjust Latch Options menu.

<table>
<thead>
<tr>
<th>01/25/02</th>
<th>OpenEdge Monitor (R&amp;D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:34:27</td>
<td>Adjust Latch Options</td>
</tr>
<tr>
<td></td>
<td>1. Spins before timeout 36000</td>
</tr>
</tbody>
</table>

Figure 20–52: PROMON Adjust Latch Options menu

This menu contains the Spins before timeout option, which 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 the “Spin locks” section on page 14–39.

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.) Figure 20–53 shows a sample Adjust Page Writer Options menu.

<table>
<thead>
<tr>
<th>01/25/02</th>
<th>OpenEdge Monitor (R&amp;D)</th>
</tr>
</thead>
<tbody>
<tr>
<td>16:19:34</td>
<td>Adjust Page Writer Options</td>
</tr>
<tr>
<td></td>
<td>1. APW queue check time: 100 milliseconds</td>
</tr>
<tr>
<td></td>
<td>2. APW buffers scan time: 1 seconds</td>
</tr>
<tr>
<td></td>
<td>3. APW buffers per scan: 1</td>
</tr>
<tr>
<td></td>
<td>4. APW writes per scan: 25</td>
</tr>
</tbody>
</table>

Figure 20–53: PROMON Adjust Page Writer Options menu

Choose one of the following options:

- **APW queue check time** — Allows you to dynamically adjust the value of the Page Writer Queue Delay (-pqde1ay) parameter.

  Note: Though you can set the value of -pqde1ay, it is a self-tuning parameter. When the database encounters heavy loads, -pqde1ay decreases its value so that the APW writes more often. When the demand on the database lessens, -pqde1ay increases its value so that the APW writes less often. The default value, in milliseconds, for -pqde1ay is 100.

- **APW buffers scan time** — Allows you to dynamically adjust the value of the Page Writer Scan Delay (-pwsde1ay) parameter. The default value, in seconds, for -pwsde1ay is 1.
- **APW buffers per scan** — Enables you to dynamically adjust the value of the Page Writer Scan (\(-\text{pwscan}\)) parameter.

- **APW writes per scan** — Enables you to dynamically adjust the value of the Page Writer Maximum Buffers (\(-\text{pwwmax}\)) 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.

### Terminate a Server

Enables you to select a server for termination. **Figure 20–54** shows the Terminate a Server display.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>19952</td>
<td>Login</td>
<td>TCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>17872</td>
<td>Login</td>
<td>TCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>2054</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>Inactive</td>
<td>TCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>Inactive</td>
<td>TCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>Inactive</td>
<td>TCP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 20–54: PROMON Terminate a Server display**

### 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. **Figure 20–55** displays available settings.

**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.

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.
R&D Adjust Monitor Options

Displays parameters controlling the PROMON display and allows you to change them. Figure 20–56 shows the Adjust Monitor Options menu.

<table>
<thead>
<tr>
<th>Option Number</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Display page length</td>
<td>24 lines</td>
</tr>
<tr>
<td>2</td>
<td>Clear screen for first page</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>Monitor sampling interval</td>
<td>10 seconds</td>
</tr>
<tr>
<td>4</td>
<td>Pause between displays</td>
<td>10 seconds</td>
</tr>
<tr>
<td>5</td>
<td>Pause between screens</td>
<td>5 seconds</td>
</tr>
<tr>
<td>6</td>
<td>Number of auto repeats</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>Change working area</td>
<td>All areas</td>
</tr>
</tbody>
</table>

Figure 20–56: PROMON Adjust Monitor Options menu

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

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. Figure 20–57 shows an example of this option's output.

<table>
<thead>
<tr>
<th>Transaction Control:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usr</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>

RETURN - repeat, U - continue uninterrupted, Q - quit

Figure 20–57: Sample output for PROMON 2PC Transaction Control option

**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. When you choose this option, PROMON displays the menu shown in Figure 20–58.

<table>
<thead>
<tr>
<th>1</th>
<th>Abort a Limbo Transaction</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Commit a Limbo Transaction</td>
</tr>
<tr>
<td>Q</td>
<td>Quit</td>
</tr>
</tbody>
</table>

Enter choice>

Figure 20–58: Sample output for PROMON Resolve 2PC Limbo Transactions option

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.

Figure 20–59 shows an example of this option’s output.

```
OpenEdge MONITOR Version 10.0
Database: /users/sports1
Q. QUIT

Enter the transaction number you want to find out if committed:
```

**Figure 20–59: Sample output for PROMON Coordinator Information option**

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. Figure 20–60 displays the Resolve JTA Transactions menu.

WARNING: Committing or rolling back a JTA transaction can compromise 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:

Figure 20–60: PROMON Resolve JTA Transactions menu

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.

Figure 20–61 shows the menu that appears when you choose the Modify Defaults option.

![Figure 20–61: PROMON Modify Defaults menu](image)

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.
This chapter describes the OpenEdge database administration utility PROUTIL.
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 passwd]] [-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 21–1.

`-userid username [-password passwd]`

Specifies the userid and password of an authenticated database administrator.

`-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 Chapter 10, “Transparent Data Encryption,” 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 Chapter 19, “Database Startup Parameters,” for a description of each database-related internationalization startup parameter.

Table 21–1 lists all of the PROUTIL qualifiers. See the appropriate section for each of the qualifiers for more detailed information.
### Table 21–1: PROUTIL utility qualifiers

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2PHASE BEGIN</td>
<td>Enables two-phase commit.</td>
</tr>
<tr>
<td>2PHASE COMMIT</td>
<td>Determines whether the coordinator database committed a transaction.</td>
</tr>
<tr>
<td>2PHASE END</td>
<td>Disables two-phase commit.</td>
</tr>
<tr>
<td>2PHASE MODIFY</td>
<td>Changes the nickname or priority of a database for two-phase commit.</td>
</tr>
<tr>
<td>2PHASE RECOVER</td>
<td>Commits or aborts limbo transactions for a database.</td>
</tr>
<tr>
<td>AUDITARCHIVE</td>
<td>Archives audit data from a database to a binary file.</td>
</tr>
<tr>
<td>AUDITLOAD</td>
<td>Loads audit data from a binary file into a database.</td>
</tr>
<tr>
<td>BIGROW</td>
<td>Specifies the number of BI clusters available to a database.</td>
</tr>
<tr>
<td>BULKLOAD</td>
<td>Loads data files into a database.</td>
</tr>
<tr>
<td>BUSY</td>
<td>Determines whether the database is currently in use.</td>
</tr>
<tr>
<td>CHANALYS</td>
<td>Displays information about free chain blocks.</td>
</tr>
<tr>
<td>CODEPAGE- COMPILER</td>
<td>Compiles a conversion map file from text format to binary format.</td>
</tr>
<tr>
<td>CONV910</td>
<td>Converts a Progress Version 9 database to an OpenEdge Release 10 database.</td>
</tr>
<tr>
<td>CONVCHAR</td>
<td>Converts a database’s character set or identifies a database’s character set (for undefined databases).</td>
</tr>
<tr>
<td>CONVFILE</td>
<td>Converts a text file from one character set to any other character set.</td>
</tr>
<tr>
<td>DBANALYS</td>
<td>Displays statistical information about index, record, and free chain blocks.</td>
</tr>
<tr>
<td>DBAUTHKEY</td>
<td>Sets an authorization key for a database.</td>
</tr>
<tr>
<td>DBIPCS</td>
<td>Displays status information for shared-memory segments attached by all OpenEdge databases on the system.</td>
</tr>
<tr>
<td>DESCRIBE</td>
<td>Describes the enabled features of a database.</td>
</tr>
<tr>
<td>DISABLEAUDITING</td>
<td>Disables auditing for a database.</td>
</tr>
<tr>
<td>DISABLEB2</td>
<td>Removes an area from Alternate Buffer Pool processing.</td>
</tr>
<tr>
<td>DISABLEENCRYPTION</td>
<td>Disables transparent data encryption for a database.</td>
</tr>
<tr>
<td>DISABLEJTA</td>
<td>Disables database participation in distributed JTA transactions.</td>
</tr>
</tbody>
</table>
### Table 21–1: PROUTIL utility qualifiers

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DISABLEKEYEVENTS</td>
<td>Disables the storage of key events for a database.</td>
</tr>
<tr>
<td>DISPTOSSCREATELIMITS</td>
<td>Display the block toss and create limits for all tables and BLOBS in a specified area.</td>
</tr>
<tr>
<td>DUMP</td>
<td>Performs a binary dump, which is generally faster than an ASCII dump.</td>
</tr>
<tr>
<td>DUMPSPECIFIED</td>
<td>Performs a selective binary dump and allows you to dump by field value.</td>
</tr>
<tr>
<td>ENABLEAUDITING</td>
<td>Enables auditing for a database.</td>
</tr>
<tr>
<td>ENABLEB2</td>
<td>Designates an area for Alternate Buffer Pool processing.</td>
</tr>
<tr>
<td>ENABLEENCRYPTION</td>
<td>Enables transparent data encryption for a database.</td>
</tr>
<tr>
<td>ENABLEJTA</td>
<td>Enables the database to participate in distributed JTA transactions.</td>
</tr>
<tr>
<td>ENABLEKEYEVENTS</td>
<td>Enables the storage of key events for a database.</td>
</tr>
<tr>
<td>ENABLELARGEFILES</td>
<td>Enables large file processing for a database.</td>
</tr>
<tr>
<td>ENABLELARGEKEYS</td>
<td>Enables large key support for a database migrated to this release.</td>
</tr>
<tr>
<td>ENABLESEQ64</td>
<td>Enables 64-bit sequence support for a database migrated to this release.</td>
</tr>
<tr>
<td>ENABLESTOREDPROC</td>
<td>Enables stored procedures and triggers for 64-bit databases created prior to this release.</td>
</tr>
<tr>
<td>EPOLICY INFO</td>
<td>Displays the valid ciphers for the database master key and object-level encryption and their command line cipher number.</td>
</tr>
<tr>
<td>EPOLICY MANAGE</td>
<td>Manages encryption policies for a specified database object, or the autostart or key store status.</td>
</tr>
<tr>
<td>EPOLICY SCAN</td>
<td>Reports on the encryption policies for the specified database object, or the autostart or key store status.</td>
</tr>
<tr>
<td>EPOLICY VIEW</td>
<td>Views the encryption policies for the specified database object, or the autostart or key store status.</td>
</tr>
<tr>
<td>HOLDER</td>
<td>Determines whether the database is currently in use in single-user mode, multi-user mode, or by a utility.</td>
</tr>
<tr>
<td>IDXACTIVATE</td>
<td>Activates an inactive index.</td>
</tr>
<tr>
<td>IDXANALYS</td>
<td>Displays information on index blocks.</td>
</tr>
<tr>
<td>IDXCOMPACT</td>
<td>Performs index compaction online and increases space utilization of the index block to a specified compacting percentage.</td>
</tr>
<tr>
<td>Qualifier</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>IDXMOVE</td>
<td>Moves an index from one application data area to another while the database remains online.</td>
</tr>
<tr>
<td>IDXBUILD</td>
<td>Consolidates index records to use disk space as efficiently as possible; activates deactivated indexes in the database. Repairs corrupted indexes in the database.</td>
</tr>
<tr>
<td>IDXCHECK</td>
<td>Checks database indexes to determine whether an index is corrupt, and if it is, diagnoses the problem.</td>
</tr>
<tr>
<td>IDXFIX</td>
<td>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.</td>
</tr>
<tr>
<td>INCREASETO</td>
<td>Increase certain server startup parameters when the database is online.</td>
</tr>
<tr>
<td>IOSTATS</td>
<td>Provides current statistics for active databases. The statistics include buffered, unbuffered, and logical I/O database operations. The statistics are cumulative from database startup.</td>
</tr>
<tr>
<td>LOAD</td>
<td>Performs a binary load.</td>
</tr>
<tr>
<td>MVSCH</td>
<td>Moves schema after a database conversion.</td>
</tr>
<tr>
<td>RCODEKEY</td>
<td>Inserts the authorization key into existing CRC-based r-code.</td>
</tr>
<tr>
<td>REVERT</td>
<td>Examines the database use at this release, and if possible, reverses the database to a previous release format.</td>
</tr>
<tr>
<td>SETAREACREATELIMIT</td>
<td>Sets the create limit for the specified area.</td>
</tr>
<tr>
<td>SETAREATOSSLIMIT</td>
<td>Sets the toss limit for the specified area.</td>
</tr>
<tr>
<td>SETBLOBCREATELIMIT</td>
<td>Sets the create limit for the specified BLOB object.</td>
</tr>
<tr>
<td>SETBLOTOSSLIMIT</td>
<td>Sets the toss limit for the specified BLOB object.</td>
</tr>
<tr>
<td>SETTABLECREATELIMIT</td>
<td>Sets the create limit for the specified table.</td>
</tr>
<tr>
<td>SETTABLETOSSLIMIT</td>
<td>Sets the toss limit for the specified table.</td>
</tr>
<tr>
<td>TABANALYS</td>
<td>Displays information about the degree of fragmentation for each table in a database.</td>
</tr>
<tr>
<td>TABLEMOVE</td>
<td>Moves a table and optionally its associated indexes from one storage area to another while the database remains online.</td>
</tr>
<tr>
<td>TRUNCATE AREA</td>
<td>Truncates application data storage areas in the specified database; use this qualifier before you remove storage areas and extents.</td>
</tr>
</tbody>
</table>
Table 21–1: PROUTIL utility qualifiers

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUNCATE BI</td>
<td>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.</td>
</tr>
<tr>
<td>UPDATESCHEMA</td>
<td>Loads the specified database with the most recent meta-schema.</td>
</tr>
<tr>
<td>UPDATEVST</td>
<td>Loads the specified database with the most recent virtual system tables.</td>
</tr>
<tr>
<td>VIEWB2</td>
<td>Lists the assignment of areas and objects with respect to Alternate Buffer Pool processing</td>
</tr>
<tr>
<td>WBREAK- COMPILER</td>
<td>Compiles the word-break table.</td>
</tr>
<tr>
<td>WORD-RULES</td>
<td>Compiles a word-rules file.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Syntax</th>
</tr>
</thead>
</table>

**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:

<table>
<thead>
<tr>
<th>If your date range is . . .</th>
<th>Then the data archived is . . .</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blank</td>
<td>All audit data records in the database at the time the archive process begins.</td>
</tr>
<tr>
<td>One date</td>
<td>All audit data records in the database with a date equal to or earlier than the specified date.</td>
</tr>
<tr>
<td>Two dates, separated by a space</td>
<td>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.</td>
</tr>
</tbody>
</table>

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*.

-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* 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 6

For an Enterprise database enabled for Transparent Data Encryption, specify -Cipher 6 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.

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 6` and 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

<table>
<thead>
<tr>
<th>Operating system</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX Windows</td>
<td><code>proutil db-name -c auditload archive-file-name [-recs num-recs] [-checkseal] [-userid user-id [-password passwd]] [-Cipher 6]</code></td>
</tr>
</tbody>
</table>

\textit{db-name}

Specifies the database where the archived data is being loaded.

\textit{archive-file-name}

Specifies the audit archive file to load.

\textit{-recs num-recs}

Specifies the number of records to load in a single transaction. AUDITLOAD executes a transaction for every \textit{num-recs} records it loads. The default for \textit{num-recs} is 100, and the maximum is your current Lock Table Entries (-L) setting. The multi-user default for -L is 8192.

\textit{-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 \textit{OpenEdge Getting Started: Core Business Services}. 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.

\textit{-userid username}

Specifies the user name of the privileged user. If you are using the \_User table, \textit{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, \textit{username} specifies a local operating system login ID. Regardless of the source, \textit{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 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 6

For an Enterprise database enabled for Transparent Data Encryption, specify -Cipher 6 to indicate that the contents of the archive file being loaded is encrypted. You are prompted to enter the passphrase that was specified when the archive file was created.

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 6 when created, you must specify -Cipher 6 for PROUTIL AUDITLOAD to indicate that the file is encrypted. You are prompted to enter the passpharse 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 the “PROUTIL EPOLICY MANAGE qualifier” section on page 21–59 or Chapter 10, “Transparent Data Encryption,” for more information on creating an encryption policy.
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 Chapter 14, “Managing Performance,” 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.

**Syntax**

```
proutil db-name [-yy n] -C BULKLOAD fd-file [-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.

- **-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 Chapter 16, “Dumping and Loading,” 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 the “Auditing impact on database utilities” section on page 9–16.
PROUTIL BUSY qualifier

Determines if the database is currently in use.

```
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:

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Database is not in use</td>
</tr>
<tr>
<td>6</td>
<td>Database is in use</td>
</tr>
<tr>
<td>64</td>
<td>Database is in process of starting up</td>
</tr>
</tbody>
</table>

**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 [ $? != 0 ]
   then
      echo "Do you want to use 'proshut' to force users off the system?"
      read ans
      if [ "$ans" = y ]
         then
            proshut -by mydb
         else
            echo "Backup will not be performed."
            exit
      fi
   fi
else
   echo "Beginning backup."
fi
# Backup procedure
```
PROUTIL CHANALYS qualifier

Displays information about free and 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
```

Parameters  

`db-name`

Specifies the database you are using.
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 CONV910 qualifier

Converts a Progress Version 9 database to an OpenEdge Release 10 database.

Syntax

```
proutil db-name -C conv910
```

Parameters

`db-name`

Specifies the source database name.

Notes

- Before performing the conversion:
  - Back up your Progress Version 9 database using a Version 9 backup utility
  - Disable after-imaging
  - Disable two-phase commit
  - Truncate your BI file

- 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, revert to the backup copy of the Version 9 database and begin the conversion again.

- After completing the conversion, back up your OpenEdge Release 10 database.

- See the “Converting a Progress Version 9 database to OpenEdge Release 10” section on page 1–17 for complete instructions for converting a Progress Version 9 database to an OpenEdge Release 10 database.
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:

<table>
<thead>
<tr>
<th>Charscan searching for utf-8 character: 49854 0xc2be. (6570)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Charscan searching for utf-8 character: 49855 0xc2bf. (6570)</td>
</tr>
<tr>
<td>Charscan searching for utf-8 character: 50048 0xc380. (6570)</td>
</tr>
<tr>
<td>Charscan searching for utf-8 character: 50049 0xc381. (6570)</td>
</tr>
<tr>
<td>Charscan searching for utf-8 character: 50050 0xc382. (6570)</td>
</tr>
</tbody>
</table>

If chars can 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
PROUTIL CONVFILE qualifier

- `cnnul1.dat` — Performs no conversion; you might want to use this file as a template file for creating new conversion tables

- `i12-1250.dat` — Converts from iso8859-2 to code page 1250

- `i12-852.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 DBANALYS qualifier

Displays statistical information about index, record, and free chain blocks.

Syntax

```
proutil db-name -C dbanalys
```

Parameters

`db-name`

Specifies the database you are using.

Example

The following output is a sample display of PROUTIL DBANALYS:

```
SUMMARY FOR AREA "Inventory" : 8
------------------------------------------
<table>
<thead>
<tr>
<th>NAME</th>
<th>Records</th>
<th>Indexes</th>
<th>Combined</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUB.Bin</td>
<td>26.3K</td>
<td>16.4K</td>
<td>42.8K</td>
</tr>
<tr>
<td>PUB.InventoryTrans</td>
<td>3661B</td>
<td>0B</td>
<td>3661B</td>
</tr>
<tr>
<td>PUB.Item</td>
<td>7662B</td>
<td>8117B</td>
<td>15779B</td>
</tr>
<tr>
<td>PUB.POLine</td>
<td>217.5K</td>
<td>51.9K</td>
<td>269.4K</td>
</tr>
<tr>
<td>PUB.PurchaseOrder</td>
<td>68.6K</td>
<td>19.0K</td>
<td>87.6K</td>
</tr>
<tr>
<td>PUB.Supplier</td>
<td>1177B</td>
<td>97B</td>
<td>1274B</td>
</tr>
<tr>
<td>PUB.SupplierItemXr</td>
<td>1118B</td>
<td>1166B</td>
<td>2284B</td>
</tr>
<tr>
<td>PUB.Warehouse</td>
<td>1288B</td>
<td>440B</td>
<td>1728B</td>
</tr>
</tbody>
</table>

Size key:
B = bytes
K = kilobytes
M = megabytes
G = gigabytes
```
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:

<table>
<thead>
<tr>
<th>Task</th>
<th>old-key value</th>
<th>new-key value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the authorization key</td>
<td>Plus sign (+)</td>
<td>Authorization key</td>
</tr>
<tr>
<td>Change the authorization key</td>
<td>Current authorization key</td>
<td>New authorization key</td>
</tr>
<tr>
<td>Remove the authorization key</td>
<td>Current authorization key</td>
<td>Plus sign (+)</td>
</tr>
</tbody>
</table>

- 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 DESCRIBE qualifier

Displays descriptive information about the database and enabled features.

Syntax

```plaintext
proutil dbname -C describe
```

Parameters `dbname`

Specifies the database to describe.

PROUTIL DESCRIBE provides a basic description of the database and details which features are enabled for the database.

Figure 21–1 provides a sample of the PROUTIL DESCRIBE output.

```
OpenEdge Database Description

Database Name               : /usr1/dbs/sample_db
Version                     : 150.0
Block Size                  : 8192
Largest Cluster             : 64
Create Date                 : Fri Jul 10 13:04:58 2009
Last Open Date              : Fri Jul 10 13:25:24 2009
Prior Open Date             : Fri Jul 10 13:25:24 2009
Schema Change Date          : Fri Jul 10 13:05:09 2009

Before Imaging information
Block Size                : 8192
Cluster Size (16K Units)  : 32
Last Open Date            : Fri Jul 10 13:05:25 2009
Bi Encryption             : Enabled

After Imaging Information
Block Size                : 8192
Begin Date                : Fri Jul 10 13:05:25 2009
Last AIMAGE NEW           : Fri Jul 10 13:05:26 2009
Currently Busy Area       : 13
Current File Number       : 2
Ai Encryption             : Enabled

Backup Information
Last Full Backup Date     : Fri Jul 10 13:05:17 2009
Last Incremental Backup   : *** Not yet performed ***

Database Features

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
<th>Active</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OpenEdge Replication</td>
<td>Yes</td>
<td>Source database</td>
</tr>
<tr>
<td>9</td>
<td>64 Bit DBKEYS</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>Large Keys</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>64 Bit Sequences</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>Encryption</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>
```

Figure 21–1: PROUTIL DESCRIBE sample output
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 Open Date** — 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.
Table 21–2 lists the possible ID and feature names.

**Table 21–2: Database features**

<table>
<thead>
<tr>
<th>ID</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OpenEdge Replication</td>
</tr>
<tr>
<td>3</td>
<td>DataXtend Remote Edition</td>
</tr>
<tr>
<td>5</td>
<td>Large File support</td>
</tr>
<tr>
<td>6</td>
<td>Database Auditing</td>
</tr>
<tr>
<td>7</td>
<td>JTA</td>
</tr>
<tr>
<td>8</td>
<td>AI Management</td>
</tr>
<tr>
<td>9</td>
<td>64-bit DB Keys</td>
</tr>
<tr>
<td>10</td>
<td>Large Keys (for indexes)</td>
</tr>
<tr>
<td>11</td>
<td>64-bit Sequences</td>
</tr>
<tr>
<td>12</td>
<td>DataXtend Integration Edition</td>
</tr>
<tr>
<td>13</td>
<td>Transparent Data Encryption</td>
</tr>
<tr>
<td></td>
<td>Failover Clusters</td>
</tr>
<tr>
<td></td>
<td>Save Key Events</td>
</tr>
</tbody>
</table>

**Note** DESCRIBE can be run against a database that is online or offline.
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:

<table>
<thead>
<tr>
<th>ID</th>
<th>ShMemVer</th>
<th>Seg#</th>
<th>InUse</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>68</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>(not PROGRESS)</td>
</tr>
<tr>
<td>100</td>
<td>3</td>
<td>0</td>
<td>Yes</td>
<td>/db/work5/sports</td>
</tr>
<tr>
<td>101</td>
<td>3</td>
<td>1</td>
<td>-</td>
<td>/db/work5/sports</td>
</tr>
<tr>
<td>120</td>
<td>3</td>
<td>0</td>
<td>No</td>
<td>/db/work5/test</td>
</tr>
</tbody>
</table>

**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 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 the “Auditing states” section on page 9–6.

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 DISABLEB2**

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 DISABLEENCRYPTION

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:
  a. BI encryption is disabled (if run offline)
  b. AI encryption is disabled
  c. All ciphers of current policies are set to NULL
d. Add encrypted data are decrypted

e. All encryption policies are removed

f. 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.

Note

The database must be offline when disabling for JTA transactions.
PROUTIL DISABLEKEYEVENTS

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 DISPTOSSCREATELIMITS qualifier

Displays toss and create limits for the specified area.

Syntax

```
proutil db-name -C disptosscreatelimits area-number
```

Parameters

- `db-name`
  
  Name of the database to have toss and create limits displayed.

- `area-number`
  
  The number of the area to have toss and create limits displayed.

Note

The database must be offline when displaying create and toss limits.
PROUTIL DUMP qualifier

Performs a binary dump of a database table to a file.

```
proutil db-name -C dump [owner-name.]table-name directory [ -index num ]
{[ -thread n ] [ -threadnum nthreads ] [ -dumplist dumpfile ] }
[ -Cipher 6 ]
```

**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.

*-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*

For a threaded dump, 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 6*

For an Enterprise database enabled for Transparent Data Encryption, specify `-Cipher 6` 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.

**Notes**

- See Chapter 16, “Dumping and Loading,” 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.

- 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. 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.bdn`. Use the -dumpfile option to generate a list of the files created by the threaded dump.

- If the file specified by the -dumpfile option exists, PROUTIL DUMP will overwrite the existing file.
• 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 the “Dumping user table contents with a Data tool” section on page 16–16.

• You cannot use DUMP to dump protected audit data. For more information on auditing and utility security and restrictions, see the “Auditing impact on database utilities” section on page 9–16.

• 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 `-Cipher 6` 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 [-preferidx index-name]
   [-Cipher 6]
```

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.

- **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

- **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.

-preferidx index-name

By default, the DUMPSPECIFIED uses the primary index to sort rows. Specifying a different index with -preferidx orders the rows by the specified index.

-Cipher 6

For an Enterprise database enabled for Transparent Data Encryption, specify -Cipher 6 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 the “Auditing impact on database utilities” section on page 9–16.
- 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 -Cipher 6 and entering a passphrase for the PBE cipher. You will be prompted to enter the passphrase when loading the contents of the dump file.
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 the “Auditing states” section on page 9–6.
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 the “Auditing tables” section on page 9–10 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.
PROUTIL ENABLEB2

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 the “PROUTIL DISABLEB2” section on page 21–36.
- To view the status of areas, see the “PROUTIL VIEWB2” section on page 21–109.
- 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.
Enables your database for transparent data encryption.

**Syntax**

```bash
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 the “`PROUTIL EPOLICY INFO` qualifier” section on page 21–58 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.

-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 Table 21–3.

**Table 21–3: Passphrase constraints**

<table>
<thead>
<tr>
<th>Rule</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimum number of characters</td>
<td>8</td>
</tr>
<tr>
<td>Maximum number of characters</td>
<td>1024</td>
</tr>
<tr>
<td>Minimum number of numeric characters</td>
<td>1</td>
</tr>
<tr>
<td>Minimum number of alpha characters</td>
<td>2</td>
</tr>
<tr>
<td>Minimum number of punctuation characters</td>
<td>1</td>
</tr>
<tr>
<td>Character set</td>
<td>[a-zA-Z0-9]!@#$%^&amp;*()-{}[]|,./&lt;&gt;?;:&lt;space&gt;</td>
</tr>
<tr>
<td>First character</td>
<td>(see Character set)</td>
</tr>
<tr>
<td>Mixed case alpha required</td>
<td>True</td>
</tr>
<tr>
<td>Case sensitive</td>
<td>True</td>
</tr>
</tbody>
</table>

**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 the “PROUTIL EPOLICY MANAGE qualifier” section on page 21–59 for command syntax and details.

- If after-imaging is enabled, ENABLEENCRYPTION causes and extent switch.
PROUTIL ENABLEJTA qualifier

Enables the database to participate in JTA transactions.

Syntax

```bash
proutil db-name -C enablejta
```

Parameters

`db-name`

Name of the database to be enabled for JTA transactions.

Notes

- The database must be offline when enabling for JTA transactions.
- Databases enabled for JTA transactions must perform crash recovery in multi-user mode.
- Enabling the database for JTA transactions disables after-imaging. You must re-enable after-imaging after enabling the database for JTA transactions.
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 ENABLELARGEFILES qualifier

Enables large file processing for a database.

Syntax

```
proutil db-name -C EnableLargeFiles
```

Parameters

- `db-name`
  
  Name of the database to enable for large files.

Notes

- The database must be offline when enabling large file processing.
- There is no corresponding command to disable large file processing for a database.
- Large files are only available on Enterprise systems.
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)
```

- The database may be online or offline when enabling large keys.
- If the database is online when enabling large keys, only newly connected clients will be able to perform operations with large keys.
- A database with large keys enabled, cannot be reverted to Release 10.1A format with PROUTIL REVERT.
- 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 ENABLESEQ64 qualifier

Enables 64-bit sequences for a database migrated to 10.1C from a prior release.

Syntax

```
proutil db-name -C enableseq64
```

Parameters

`db-name`

Name of the database to be enabled for large files.

Notes

- The database may be online or offline when enabling 64-bit sequences.
- Databases created with Release 10.1C have 64-bit sequences by default.
- 64-bit sequence support cannot be disabled.
- A database with 64-bit sequence support enabled, cannot be reverted to Release 10.1A format with PROUTIL REVERT.
- Existing sequences with the upper limit specified as the Unknown value (?) were previously bounded by the maximum of a signed 32-bit integer. They are now bounded by the maximum of a signed 64-bit integer.
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 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
  7      RC4      ECB    128     binary

----------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
  7      RC4      ECB    128     binary
```
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 | adminphrase}]}
  [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.

---

**Caution:** 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 Chapter 10, “Transparent Data Encryption” and OpenEdge Getting Started: Core Business Services.
**PROUTIL EPOLICY SCAN qualifier**

Reports on the encryption policies for the specified database object, or the autostart or key store status.

**Syntax**

```plaintext
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.

`index index-name`

Specifies to view the policies of the `index-name` index.

`lob lob-name`

Specifies to view the policies of the `lob-name` lob.

`table table-name`

Specifies to view the policies of the `table-name` table.

`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.
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:

<table>
<thead>
<tr>
<th>Return code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>The database is not in use</td>
</tr>
<tr>
<td>14</td>
<td>The Database is locked by single user, PROUTIL, or RFUTIL</td>
</tr>
<tr>
<td>16</td>
<td>The database is open in multi-user mode</td>
</tr>
</tbody>
</table>

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]
```

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.

- `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.

Notes

- IDXACTIVATE works for both online and offline databases.

- If IDXACTIVATE detects clients with an earlier schema timestamp than the index, you may wait for those clients to disconnect 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.

- 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
```

Parameters

`db-name`

Specifies the database to analyze.

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 Chapter 14, “Managing Performance,” for more information.

Example

The following output is a sample display of PROUTIL IDXANALYS:

```
INDEX BLOCK SUMMARY FOR AREA "Order" : 11
-------------------------------------------------------
Table      Index Fields Levels Blocks Size % Util Factor
PUB.BillTo custnumbillto 46 2 1 1 27B 2.7 1.0
PUB.Order  CustOrder     48 2 2 44 39.1K 91.4 1.5
          OrderDate     49 1 2 33 18.5K 57.7 2.1
          OrderNum      47 1 2 38 35.3K 95.5 1.5
          OrderStatus   50 1 2 7 3808B 54.6 2.1
          SalesRep      51 1 2 9 6226B 69.5 1.9
PUB.OrderLine itemnum     23 1 2 56 36.6K 67.2 1.9
           orderline   22 2 3 139 132.7K 98.2 1.0
           OrderLineStatus 24 1 2 10 73548B 73.8 1.8
PUB.ShipTo  custnumshipto 45 2 1 1 36B 3.6 1.0
```

21–65
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.

```
proutil db-name -C idxbuild
[ all | table [owner-name.]table-name | area area-name |
  schema schema-owner | activeindexes | inactiveindexes]
[ -thread n ] [ -threadnum n ]
[ -T dir-name | -SS sort-file-directory-specification ]
[ -TB blocksize ] [ -TM n ] [ -B n ] [ -SG n ] [ -pfactor n ]
```

**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.

*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.

*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.

*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.
-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.

-T \( \text{dir-name} \)

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 \( \text{sort-file-directory-specification} \)

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.

-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 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 \((-\text{TM} + 1) \times -\text{TB}\).

-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.
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 Activation (v/V) - Rebuild selected active or inactive indexes
Quit (q/Q) - Quit, do not rebuild
```
Enter your selection:

Table 21–4 describes PROUTIL IDXBUILD options.

**Table 21–4: PROUTIL IDXBUILD options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Prompts you to verify whether you have enough disk space for index sorting.</td>
</tr>
<tr>
<td>Some</td>
<td>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.</td>
</tr>
<tr>
<td>By Area</td>
<td>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.</td>
</tr>
<tr>
<td>By Schema</td>
<td>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.</td>
</tr>
<tr>
<td>By Table</td>
<td>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.</td>
</tr>
<tr>
<td>By Activation</td>
<td>Prompts you to chose active or inactive indexes, then prompts you for the indexes, then prompts you to verify whether you have enough disk space for index sorting.</td>
</tr>
<tr>
<td>Quit</td>
<td>Quits without rebuilding any indexes.</td>
</tr>
</tbody>
</table>

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 Chapter 14, “Managing Performance,” for more information about using the IDXBUILD qualifier.

- 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.

- Use IDXFIX to repair and IDXCOMPACT to compress online.
PROUTIL Utility

PROUTIL IDXCHECK qualifier

Checks database indexes to determine whether an index is corrupt, and if it is, diagnoses the problem.

```
proutil db-name -C idxcheck [ all | table [owner-name.]table-name | area area-name | schema schema-owner]
```

**Parameters**

*db-name*

Specifies the database whose index you are checking.

*all*

Specifies that you want to check all your indexes.

*table [owner-name.]table-name*

Specifies that you want to check the indexes defined for the named 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.

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
----------------------
  Validation (o/O) - Change validation options
  Quit     (q/Q) - Quit, do not Check
```

Enter your selection:
Table 21–5 describes the PROUTIL IDXCHECK options.

**Table 21–5: PROUTIL IDXCHECK options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Checks all the indexes</td>
</tr>
<tr>
<td>Some</td>
<td>Prompts you for the indexes you want to check</td>
</tr>
<tr>
<td>By Area</td>
<td>Prompts you for the area containing the indexes you want to check</td>
</tr>
<tr>
<td>By Schema</td>
<td>Prompts you for the schema owner of the indexes you want to check</td>
</tr>
<tr>
<td>By Table</td>
<td>Prompts you for the table containing the indexes you want to check</td>
</tr>
<tr>
<td>Validation</td>
<td>Change the validation options</td>
</tr>
<tr>
<td>Quit</td>
<td>Quits without checking any indexes</td>
</tr>
</tbody>
</table>

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
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.
Table 21–6 describes the PROUTIL IDXCHECK VALIDATION options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Validates physical consistency of index blocks</td>
</tr>
<tr>
<td>2</td>
<td>Validates keys for each record</td>
</tr>
<tr>
<td>3</td>
<td>Validates record for each key</td>
</tr>
<tr>
<td>4</td>
<td>Validates key order</td>
</tr>
<tr>
<td>L</td>
<td>Locks the tables during the check</td>
</tr>
<tr>
<td>R</td>
<td>Resets the error limit; this value is set to 500</td>
</tr>
<tr>
<td>C</td>
<td>Continues to execute the check</td>
</tr>
<tr>
<td>Quit</td>
<td>Quits without checking any indexes</td>
</tr>
</tbody>
</table>

**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.
- 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 Chapter 15, “Maintaining Database Structure,” 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.
PROUTILIDXCOMPACT qualifier

Performs index compaction online.

```
proutil db-name -C idxcompact [ owner-name.]table-name.index-name [ 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.

`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.
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 Chapter 15, “Maintaining Database Structure,” for a description of how to monitor the progress of this utility using the _UserStatus virtual system table (VST).
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.

```
proutil db-name -silent -C idxfix -NL [-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.

- **-NL**
  - Avoids issuing record locks and index admin locks.

- **-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.
   2. Scan indexes for invalid index entries.
   3. Both 1 and 2 above.
   5. Build indexes from existing indexes.
   6. Delete one record and its index entries.
   7. Quit

   Enter your selection:
```
Table 21–7 describes the PROUTIL IDXFIX options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Scans the database records for missing or incorrect indexes.</td>
</tr>
<tr>
<td>2</td>
<td>Scans the index for corrupt index entries. You can specify whether to scan all indexes or a specific set of indexes.</td>
</tr>
<tr>
<td>3</td>
<td>Checks the index entries, then checks the database entries.</td>
</tr>
</tbody>
</table>
| 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) |
| 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. |
| 7      | Ends the PROUTIL Index Fix utility. |
If you specify option 1, 2, or 3 in Table 21–7, 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 Activation (v/V) - Fix selected active or inactive indexes
Quit (q/Q) - Quit, do not fix

Enter your selection:

Table 21–8 describes the PROUTIL IDXFIX options.

Table 21–8: PROUTIL IDXFIX scan options

<table>
<thead>
<tr>
<th>Option</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>All</td>
<td>Prompts you to verify whether you fix all indexes</td>
</tr>
<tr>
<td>Some</td>
<td>Prompts you for the indexes you want to fix by first entering the table name, and then the index name; IDXFIX then prompts you to verify the action</td>
</tr>
<tr>
<td>By Area</td>
<td>Prompts you for the area containing the indexes you want to fix, and then prompts you for the indexes in the area</td>
</tr>
<tr>
<td>By Schema</td>
<td>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</td>
</tr>
<tr>
<td>By Table</td>
<td>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</td>
</tr>
<tr>
<td>By Activation</td>
<td>Prompts you to chose active or inactive indexes, then prompts you for the indexes, then prompts you to verify the action</td>
</tr>
<tr>
<td>Quit</td>
<td>Quits without fixing any indexes</td>
</tr>
</tbody>
</table>

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 Chapter 15, “Maintaining Database Structure,” 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 the “Auditing impact on database utilities” section on page 9–16.

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.
PROUTIL IDXMOVE qualifier

Moves an index from one application data area to another while the database remains online.

Syntax

```bash
proutil db-name -C idxmove [ owner-name.]table-name.index-name area-name
```

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.”

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 killed, and all the blocks of the old index are being 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.

• The _UserStatus virtual system table displays the utility’s progress. See Chapter 15, “Maintaining Database Structure,” 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 -aibusf n -Mxs 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.

```
-aibusf
```

Use -aibusf to specify the new number of after-image buffers.

```
-Mxs
```

Use -Mxs to specify the size of the shared-memory overflow in kilobytes.

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 -aibusf 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 Chapter 19, “Database Startup Parameters” for more information on range and limits of -B, -L, -bibufs, -aibusf, 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 active databases.

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 IOSTAT include buffered, unbuffered, and logical I/O database operations. The statistics are cumulative from database startup.

Example

The following output is a sample display of PROUTIL IOSTATS:

```
<table>
<thead>
<tr>
<th>FILE</th>
<th>BUFFERED</th>
<th>UNBUFFERED</th>
<th>LOGICAL (2556)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Writes</td>
<td>Reads</td>
<td>Writes</td>
</tr>
<tr>
<td>----------</td>
<td>------------</td>
<td>------------</td>
<td>----------------</td>
</tr>
<tr>
<td>sports.b1</td>
<td>0</td>
<td>317</td>
<td>389</td>
</tr>
<tr>
<td>sports.d1</td>
<td>1</td>
<td>11</td>
<td>0</td>
</tr>
<tr>
<td>sports_7.d1</td>
<td>773</td>
<td>6792</td>
<td>0</td>
</tr>
<tr>
<td>sports_7.d2</td>
<td>270</td>
<td>1702</td>
<td>10</td>
</tr>
<tr>
<td>sports_8.d1</td>
<td>649</td>
<td>715</td>
<td>0</td>
</tr>
<tr>
<td>sports_8.d2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>sports_9.d1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>sports_9.d2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>sports_10.d1</td>
<td>1</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>sports_10.d2</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
```

FILE

Displays the name of the file where the statistics are displayed. The file types can include: database files (.db extensions), Before-image files (.bi extensions), After-image files (.ai extensions), and application data extents (.dn 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.
- 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 [-dumplist dumpfile ]
   build indexes [-TB blocksize] [-TM n]
   [-T dir-name] [-SS sort-file-directory-specification]
   |-Cipher 6
```

**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.

*-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 the “Loading table contents in binary format with PROUTIL” section on page 16–20.

*build indexes*

Indicates that PROUTIL LOAD will simultaneously build the indexes and perform the load.

*-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.
For an Enterprise database enabled for Transparent Data Encryption, specify -Cipher 6 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.

**Notes**

- See Chapter 16, “Dumping and Loading,” 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.

- You cannot use LOAD to load protected audit data. For more information on auditing and utility security and restrictions, see the “Auditing impact on database utilities” section on page 9–16.
• If the contents of the binary dump file was encrypted with `-Cipher 6` when created, you must specify `-Cipher 6` for PROUTIL LOAD 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 binary dump file, the loaded records are not encrypted unless there is an active encryption policy on the table before the load begins. See the “PROUTIL EPOLICY MANAGE qualifier” section on page 21–59 or Chapter 10, “Transparent Data Encryption,” for more information on creating an encryption policy.
PROUTILITY Utility

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 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:

<table>
<thead>
<tr>
<th>Task</th>
<th>old-key value</th>
<th>new-key value</th>
<th>Files</th>
</tr>
</thead>
<tbody>
<tr>
<td>Set the authorization</td>
<td>Plus sign (+)</td>
<td>Authorization key</td>
<td>R-code files</td>
</tr>
<tr>
<td>key</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Change the authorization key</td>
<td>Current authorization key</td>
<td>New authorization key</td>
<td>R-code files</td>
</tr>
<tr>
<td>Remove the authorization key</td>
<td>Current authorization key</td>
<td>Plus sign (+)</td>
<td>R-code files</td>
</tr>
</tbody>
</table>

- Once you set the authorization key, do not forget it. You cannot change or remove the authorization key without knowing its current value.
PROUTIL REVERT qualifier

Reverts the database from to a Release 10.1A format.

Syntax

```
proutil db-name -C revert
```

Parameters

`db-name`

The name of the database to revert. You cannot revert a database created with Release 10.1C. You can only revert a database migrated from a prior release.

Caution: PROUTIL REVERT is a non-recoverable utility. If PROUTIL REVERT terminates abnormally, you must restore your database from backup.

The large database features introduced in Release 10.1B require at least one manual step (running this utility) to revert a database to the 10.1A format. PROUTIL REVERT will analyze the database and determine if it can be reverted with this utility. If PROUTIL REVERT cannot revert the database, a series of manual steps, including dumping and loading the data, is required to revert the database to a 10.1A format.

PROUTIL REVERT cannot revert a database if any of the following conditions exist:

- The schema contains any INT64 fields
- The database has enabled support for large key entries for indexes
- The database has enabled support for 64-bit sequences
- The database has a Type II area with a high water mark utilizing 64-bit DB Keys
- The database has a LOB with segments utilizing 64-bit block values
- The database has area numbers greater than 1000
PROUTIL REVERT processes as follows:

1. Determines that the user has sufficient privilege to execute the command. The privilege check is limited to file system access to the database.

2. Analyzes the features of the database to determine if the database can be reverted by the utility. If not, the utility issues messages indicating why the database cannot be reverted, and exits.

The following sample output is from an attempt to revert a database that does not meet the reversion requirements:

<table>
<thead>
<tr>
<th>Feature</th>
<th>Enabled</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 Bit DBKEYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Large Keys</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>64 Bit Sequences</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Revert: The database actively supports Large Key Indexes. (13749)
Revert: The database can not be reverted. (13750)
Revert: The database contains 64-bit Sequences. (13751)
Revert: The database can not be reverted. (13750)

3. Prompt the user to confirm that the database has been backed up.

4. Perform the physical fixes necessary to revert the database. Fixes include the following:
   - Reverting values in the database master block to 10.1A format
   - Reverting object block values to 10.1A format
   - Removing 10.1C VSTs and adding the 10.1A VSTs
   - Removing all 10.1C settings from the feature mask

5. Disable 2-Phase commit, JTA, Replication, After-Imaging, and AI Management, if active.

6. Truncate the BI file.
PROUTIL REVERT provides informational messages updating the progress of the revert process. This following sample output is from a successful revert:

```
Revert Utility
---------------------
<table>
<thead>
<tr>
<th>Feature</th>
<th>Enabled</th>
<th>Active</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Auditing</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>64 Bit DBKEYS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Revert: Have you backed up your database..... Type y to continue (y/n).
(y)
Revert: Database beginning reversion process. (13754)
VST Table Deletion has begun. Please standby. (6876)
VST Table Deletion has completed successfully. (6885)
Revert: 64 Bit Dbkey has been disabled for this database.
After-image disabled. (846)
Revert: Truncating the Bi File.
VST Table Deletion has begun. Please standby. (6876)
VST Table Deletion has completed successfully. (6885)
Adding VST file: _Connect. (6875)
Adding VST file: _MstrBlk. (6875)
Adding VST file: _DbStatus. (6875)
.
.
Adding VST file: _AreaThreshold. (6875)
Total number of Virtual System Tables is 45. (6250)
Revert: Database ending reversion process.
```

When PROUTIL REVERT completes successfully, the database is in 10.1A format. You should perform a backup with 10.1A; previous backups in 10.1C format are incompatible with the reverted database.

**Notes**

- PROUTIL REVERT must be run against an offline database.
- PROUTIL REVERT expects to find the 10.1A database utilities directory, 101dbutils, in your installation bin directory. If the 10.1A utility directory is not found, PROUTIL REVERT cannot rebuild the 10.1A VSTs, and issues a message. You must then rebuild the VSTs with the 10.1A version of PROUTIL UPDATEVST.
**PROUTIL SETAREACREATELIMIT qualifier**

Set the create limit for the specified area.

**Syntax**

```
proutil db-name -C setareacreatelimit area-number create-limit
```

**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.

- **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.

**Notes**

- SETAREACREATELIMIT will set the limit for all tables and BLOBs in the specified area.

- The create limit value must be 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.

- The database must be offline when setting create limits.

- For more information on setting create limits, see the “Database fragmentation” section on page 14–40.
PROUTIL SETAREATOSSLIMIT qualifier

Set the toss limit for the specified area.

Syntax

```
proutil db-name -C setareatosslimit area-number toss-limit
```

Parameters

- `db-name`
  Specifies the database where you are setting the toss limit.

- `area-number`
  Specifies the area where you are setting the toss limit.

- `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.

Notes

- SETAREATOSSLIMIT will set the limit for all tables and BLOBs 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.
- The database must be offline when setting toss limits.
- For more information on setting toss limits, see the “Database fragmentation” section on page 14–40.
**PROUTIL SETBLOBCREATELIMIT** qualifier

Set the create limit for the specified BLOB.

**Syntax**

```
proutil db-name -C setblobcreatelimit blob-id create-limit
```

**Parameters**

- `db-name`
  Specifies the database where you are setting the create limit.

- `blob-id`
  Specifies the BLOB object for which you are setting the create limit.

- `create-limit`
  Specifies the new create limit.

**Notes**

- For a Type II area, SETBLOBCREATELIMIT sets the BLOB create limit for the specified BLOB. For a Type I area, SETBLOBCREATELIMIT issues a warning that all tables and BLOBs in an area must have the same create limit, and prompts you to confirm setting the create limit for all tables and BLOBs in the area containing the specified BLOB.

- The create limit value must be 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.

- The database must be offline when setting create limits.

- For more information on setting create limits, see the “Database fragmentation” section on page 14–40.
PROUTIL Utility

**PROUTIL SETBLOBTOSSLIMIT qualifier**

Set the toss limit for the specified BLOB.

**Syntax**

```
proutil db-name -C setblobtosslimit blob-id toss-limit
```

**Parameters**

`db-name`

Specifies the database where you are setting the toss limit.

`blob-id`

Specifies the BLOB object for which you are setting the toss limit.

`toss-limit`

Specifies the new toss limit.

**Notes**

- For a Type II area, SETBLOBTOSSLIMIT sets the BLOB toss limit for the specified BLOB. For a Type I area, SETBLOBTOSSLIMIT issues a warning that all tables and BLOBs in an area must have the same toss limit, and prompts you to confirm setting the toss limit for all tables and BLOBs in the area containing the specified BLOB.

- 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.

- The database must be offline when setting toss limits.

- For more information on setting toss limits, see the “Database fragmentation” section on page 14–40.
PROUTIL SETTABLECREATELIMIT qualifier

Set the create limit for the specified table.

Syntax

```
proutil db-name -C settablecreatelimit table-name create-limit
```

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.

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 BLOBs in an area must have the same create limit, and prompts you to confirm setting the create limit for all tables and BLOBs in the area containing the specified table.

- The create limit value must be 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.

- The database must be offline when setting create limits.

- For more information on setting create limits, see the “Database fragmentation” section on page 14–40.
PROUTIL Utility

PROUTIL SETTABLETOSSLIMIT qualifier

Set the toss limit for the specified table.

Syntax

```
proutil db-name -C settabletosslimit table-name toss-limit
```

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.

Notes

- For a Type II area, SETTABLETOSSLIMIT sets the table toss limit for the specified table. For a Type I area, SETTABLECREATELIMIT issues a warning that all tables and BLOBs in an area must have the same toss limit, and prompts you to confirm setting the toss limit for all tables and BLOBs 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.

- The database must be offline when setting create limits.

- For more information on setting toss limits, see the "Database fragmentation" section on page 14–40.
**PROUTIL TABANALYS qualifier**

Displays information about the degree of fragmentation for each table in a database. Also displays summary information about record sizes for each table.

**Syntax**

```
proutil db-name -C tabanalys
```

**Parameters**

`db-name`

Specifies the database to display information.

**Example**

The following output is a sample display of PROUTIL TABANALYS:

<table>
<thead>
<tr>
<th>Table</th>
<th>Records</th>
<th>Size</th>
<th>Min</th>
<th>Max</th>
<th>Mean</th>
<th>Count</th>
<th>Factor</th>
<th>Scattering Factor</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUB.Benefits</td>
<td>21</td>
<td>848B</td>
<td>39</td>
<td>41</td>
<td>40</td>
<td>21</td>
<td>1.0</td>
<td>2.0</td>
</tr>
<tr>
<td>PUB.Department</td>
<td>7</td>
<td>211B</td>
<td>26</td>
<td>35</td>
<td>30</td>
<td>7</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>PUB.Employee</td>
<td>55</td>
<td>6434B</td>
<td>99</td>
<td>135</td>
<td>115</td>
<td>55</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>PUB.Family</td>
<td>72</td>
<td>3195B</td>
<td>38</td>
<td>51</td>
<td>44</td>
<td>72</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>PUB.Timesheet</td>
<td>25</td>
<td>1081B</td>
<td>42</td>
<td>45</td>
<td>43</td>
<td>25</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>PUB.Vacation</td>
<td>12</td>
<td>288B</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>12</td>
<td>1.0</td>
<td>1.0</td>
</tr>
</tbody>
</table>

**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.
- See Chapter 14, “Managing Performance,” for more information about database fragmentation.
PROUTIL TABLEMOVE qualifier

Moves a table and optionally 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]
```

**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 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.”

**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.

- 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 Chapter 15, “Maintaining Database Structure,” for more information.
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. Only a user with Audit Archiver privilege can run this utility on an area containing audit data. 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 the “Auditing impact on database utilities” section on page 9–16.

- 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.
PROUTIL Utility

- 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 Chapter 15, “Maintaining Database Structure,” 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 524K. 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.
- Running PROUTIL UPDATESCHEMA on a database from the current release informs you that your database is up to date.
PROUTIL Utility

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 Chapter 26, “Virtual System Tables,” for a list of the virtual system tables that OpenEdge provides.
PROUTIL VIEWB2

Lists the assignment of areas and objects with respect to alternate buffer pool processing.

Syntax

```
proutil db-name -C viewb2
```

Parameters

`db-name`

Name of the database where you are viewing alternate buffer pool assignments.

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.

An example of output from PROUTIL VIEWB2, follows:
OpenEdge Release 10.2B1P as of Wed Oct 21 19:01:48 EDT 2009
Buffer pool assignment for areas and objects:

Area 6: "Schema Area" - Primary Buffer Pool

<table>
<thead>
<tr>
<th>Object Enablement</th>
<th>Type</th>
<th>Object Id: Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Table</td>
<td>-208: PUB._KeyEvent</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-207: PUB._sec-authentication-domain</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-206: PUB._sec-authentication-system</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-205: PUB._sec-granted-role-condition</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-204: PUB._sec-granted-role</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-203: PUB._sec-role</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-202: PUB._Db-Option</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-201: PUB._Db-Detail</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-200: PUB._Client-Session</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-126: PUB._SQL_Properties</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-125: PUB._Sysproperties</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-124: PUB._Sysroles</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-123: PUB._Syschkconstr_Name_Map</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-122: PUB._Sysviews_Name_Map</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-121: PUB._Sysseqauth</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-120: PUB._Syststzstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-119: PUB._Sysnvarcharstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-118: PUB._Sysncharstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-117: PUB._Sysbigintstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-116: PUB._Systinyintstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-115: PUB._Systsstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-114: PUB._Sysrealstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-113: PUB._Syscharstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-112: PUB._Systimestat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-110: PUB._Sysfloatstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-109: PUB._Sysdatestat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-108: PUB._Sysnumstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-107: PUB._Sysintstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-106: PUB._Syssmintstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-105: PUB._Sysvarcharstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-104: PUB._Syscolstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-103: PUB._Sysproctext</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-102: PUB._Sysprocbin</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-99: PUB._Systrigcols</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-98: PUB._Systrigger</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-97: PUB._Sysproccolumns</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-96: PUB._Sysprocedures</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-95: PUB._Sysattachtbls</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-94: PUB._Sysidxstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-93: PUB._Systblstat</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-92: PUB._Syscalc1table</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-91: PUB._Sysdatatypes</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-90: PUB._Syschkcolusage</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-89: PUB._Systb1constrs</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-88: PUB._Syschkconstrs</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-87: PUB._Sysrefconstrs</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-86: PUB._Syskeycolusage</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-85: PUB._Sysdblinks</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-84: PUB._Syssynonyms</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-83: PUB._Syscolauth</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-82: PUB._Systabauth</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-81: PUB._Sysdbauth</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-80: PUB._Sysviews</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-79: PUB._Word-rule</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>-70: PUB._StorageObject</td>
</tr>
</tbody>
</table>
| Default Table -69: PUB._Collation
| Default Table -68: PUB._Codepage
| Default Table -67: PUB._Sequence
| Default Table -66: PUB._Field-Trig
| Default Table -65: PUB._File-Trig
| Default Table -64: PUB._View-Ref
| Default Table -63: PUB._View-Col
| Default Table -62: PUB._View
| Default Table -61: PUB._Db
| Default Table -60: PUB._User
| Default Table -59: PUB._Index-Field
| Default Table -58: PUB._Index
| Default Table -57: PUB._Field
| Default Table -56: PUB._File
| Default Index -1318: _KeyEvent-Urnum (PUB._KeyEvent)
| Default Index -1317: _KeyEvent-MsgId (PUB._KeyEvent)
| Default Index -1316: _KeyEvent-Date (PUB._KeyEvent)
| Default Index -1315: _Domain-type
| Default Index -1314: _Domain-enabled
| Default Index -1313: _Domain-description
| Default Index -1312: _Domain-name
| Default Index -1311: _PAM-module-name
| Default Index -1310: _Domain-type-description
| Default Index -1309: _Domain-type
| Default Index -1308: _Conditional-attribute
| Default Index -1307: _Granted-role-guide
| Default Index -1306: _Role-name (PUB._sec-granted-role)
| Default Index -1305: _Grantor (PUB._sec-granted-role)
| Default Index -1304: _Grantee (PUB._sec-granted-role)
| Default Index -1303: _Granted-role-guide (PUB._sec-granted-role)
| Default Index -1302: _Role-description (PUB._sec-role)
| Default Index -1301: _Role-creator (PUB._sec-role)
| Default Index -1300: _Role-name (PUB._sec-role)
| Default Index 0: _Userid (PUB._User)
| Default Index 1: _File-Name (PUB._File)
| Default Index 2: _File/Field (PUB._Field)
| Default Index 3: _Field-Name (PUB._Field)
| Default Index 4: _Field-Position (PUB._Field)
| Default Index 5: _File/Index (PUB._Index)
| Default Index 6: _Index/Number (PUB._Index-Field)
| Default Index 7: _Field (PUB._Index-Field)
| Default Index 28: default (PUB.Local-Default)
| Default Index 994: _View-Name (PUB._View)
| Default Index 995: _View-Col (PUB._View-Col)
| Default Index 996: _Vcol-Position (PUB._View-Col)
| Default Index 997: _View-Ref (PUB._View-Ref)
| Default Index 998: _Vref-View (PUB._View-Ref)
| Default Index 999: _Index-Name (PUB._Index)
| Default Index 1000: _File-Trig (PUB._File-Trig)
| Default Index 1020: _Field-Trig-DBK (PUB._Field-Trig)
| Default Index 1021: _Field-Trig-Rpos (PUB._Field-Trig)
| Default Index 1022: _Db/File (PUB._file)
| Default Index 1023: _Database (PUB._Db)
| Default Index 1024: _Dump-Name (PUB._File)
PROUTIL VIEWB2 output

(3 of 4)

| Default | Index 1025: _Seq-Name (PUB._Sequence) |
| Default | Index 1026: _Object-Id (PUB._StorageObject) |
| Default | Index 1031: _Viewidx (PUB._Sysviews) |
| Default | Index 1032: _File-Number (PUB._File) |
| Default | Index 1033: _Index-Number (PUB._Index) |
| Default | Index 1034: _Sysprvsysindx (PUB._Syssysindices) |
| Default | Index 1035: _Syspubsysindx (PUB._Syssysindices) |
| Default | Index 1036: _Idxtblconstrs (PUB._Systblconstrs) |
| Default | Index 1037: _Idxkeycolusage (PUB._Syskeycolusage) |
| Default | Index 1038: _Idxchkconstrs (PUB._Syschkconstrs) |
| Default | Index 1039: _Idxchkcolusage (PUB._Syschkcolusage) |
| Default | Index 1040: _Idxrefconstrs (PUB._Sysrefconstrs) |
| Default | Index 1041: _Idxchkcolcolusage (PUB._Syschkcolusage) |
| Default | Index 1042: _Idxrefcnstrsref (PUB._Sysrefconstrs) |
| Default | Index 1043: _Idxsystblstat (PUB._Systblstat) |
| Default | Index 1044: _Idxsysidxstat (PUB._Sysidxstat) |
| Default | Index 1045: _Idxsyscolstat (PUB._Syscolstat) |
| Default | Index 1046: _Idxsyscharstat (PUB._Syscharstat) |
| Default | Index 1047: _Idxsysdatestat (PUB._Sysdatestat) |
| Default | Index 1048: _Idxsysintstat (PUB._Sysintstat) |
| Default | Index 1049: _Idxsysnumstat (PUB._Sysnumstat) |
| Default | Index 1050: _Idxsysnumstat (PUB._Sysnumstat) |
| Default | Index 1051: _Idxsysbinstat (PUB._Sysbinstat) |
| Default | Index 1052: _Idxsysrealstat (PUB._Sysrealstat) |
| Default | Index 1053: _Idxsysstzstat (PUB._Sysstzstat) |
| Default | Index 1054: _Idxsysdbauth (PUB._Sysdbauth) |
| Default | Index 1055: _Idxsysvarcharstat (PUB._Sysvarcharstat) |
| Default | Index 1056: _Idxsyscharstat (PUB._Syscharstat) |
| Default | Index 1057: _Idxsystbllist (PUB._Systblstat) |
| Default | Index 1058: _Idxsysfloatstat (PUB._Sysfloatstat) |
| Default | Index 1059: _Idxsyscolauth (PUB._Syscolauth) |
| Default | Index 1060: _Idxsysobject (PUB._Sysobject) |
| Default | Index 1061: _Idxsysobject (PUB._Sysobject) |
| Default | Index 1062: _Idxsyscharstat (PUB._Syscharstat) |
| Default | Index 1063: _Idxsysintstat (PUB._Sysintstat) |
| Default | Index 1064: _Idxsysprocbin (PUB._Sysprocbin) |
| Default | Index 1065: _Idxsysprocedures (PUB._Sysprocedures) |
| Default | Index 1066: _Idxsysdbauth (PUB._Sysdbauth) |
| Default | Index 1067: _Idxsysstrigger (PUB._Sysstrigger) |
| Default | Index 1068: _Idxsysdblinks (PUB._Sysdblinks) |
| Default | Index 1069: _Idxsysdataypes (PUB._Sysdatatypes) |
| Default | Index 1070: _Idxsyscalctable (PUB._Syscalctable) |
| Default | Index 1071: _Idxsysattachtbls (PUB._Sysattachtbls) |
| Default | Index 1072: _Idxsysproccolumns (PUB._Sysproccolumns) |
| Default | Index 1073: _Idxsysrigrocols (PUB._Sysrigrocols) |
| Default | Index 1074: _Idxsysproctext (PUB._Sysproctext) |
| Default | Index 1075: _Idxsysstzstat (PUB._Sysstzstat) |
| Default | Index 1076: _Idxsysseqauth (PUB._Sysseqauth) |
| Default | Index 1077: _Idxsysviews (PUB._Sysviews) |
| Default | Index 1078: _Idxsyskonstrnmap (PUB._Syskonstrnmap) |

(PUB._Syskonstrnmap)

| Default | Index 1079: _Idxsysroles (PUB._Sysroles) |
| Default | Index 1080: _Idxsysobjects (PUB._Sysobjects) |
| Default | Index 1081: _Idxsqlproperties (PUB._SQLProperties) |
| Default | Index 1082: _Userid (PUB._Userid) |
| Default | Index 1083: _Auth-time (PUB._Auth-time) |
| Default | Index 1084: _Db-guid (PUB._Db-guid) |
| Default | Index 1085: _Session-uid (PUB._Session-uid) |
| Default | Index 1086: _Db-guid (PUB._Db-guid) |
| Default | Index 1087: _Fieldphyspos (PUB._Field) |
| Default | Index 1088: _Option-code (PUB._Option-code) |
| Default | Index 1089: _Codepage-Name (PUB._Codepage) |
| Default | Index 1090: _Codepage-Seq (PUB._Codepage) |
| Default | Index 1091: _Collation-Name (PUB._Collation) |
| Default | Index 1092: _Collation-Seq (PUB._Collation) |
| Default | Index 1093: _Wr-Number (PUB._Word-rule) |
### Area 7: "Info Area" - Primary Buffer Pool

<table>
<thead>
<tr>
<th>Object Enablement</th>
<th>Type</th>
<th>Object Id: Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Table</td>
<td>1: PUB.Invoice</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>3: PUB.Item</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>6: PUB.Salesrep</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>7: PUB.State</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>8: PUB.Local-Default</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>8: Invoice-Num (PUB.Invoice)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>9: Cust-Num (PUB.Invoice)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>10: Invoice-Date (PUB.Invoice)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>11: Order-Num (PUB.Invoice)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>17: Item-Num (PUB.Item)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>18: Cat-Description (PUB.Item)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>19: Item-Name (PUB.Item)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>26: Sales-Rep (PUB.Salesrep)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>27: State (PUB.State)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>29: Call-Num (PUB.Ref-Call)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>30: Cust-Num (PUB.Ref-Call)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>31: Sibling (PUB.Ref-Call)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>32: Txt (PUB.Ref-Call)</td>
</tr>
</tbody>
</table>

### Area 8: "Customer/Order Area" - Alternate Buffer Pool

<table>
<thead>
<tr>
<th>Object Enablement</th>
<th>Type</th>
<th>Object Id: Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate</td>
<td>Table</td>
<td>2: PUB.Customer</td>
</tr>
<tr>
<td>Default</td>
<td>Table</td>
<td>4: PUB.Order</td>
</tr>
<tr>
<td>Alternate</td>
<td>Table</td>
<td>5: PUB.Order-Line</td>
</tr>
<tr>
<td>Alternate</td>
<td>Index</td>
<td>16: Sales-Rep (PUB.Customer)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>21: Cust-Order (PUB.Order)</td>
</tr>
</tbody>
</table>

### Area 9: "Primary Index Area" - Primary Buffer Pool

<table>
<thead>
<tr>
<th>Object Enablement</th>
<th>Type</th>
<th>Object Id: Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate</td>
<td>Index</td>
<td>12: Cust-Num (PUB.Customer)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>20: Order-Num (PUB.Order)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>24: order-line (PUB.Order-Line)</td>
</tr>
</tbody>
</table>

### Area 10: "Customer Index Area" - Primary Buffer Pool

<table>
<thead>
<tr>
<th>Object Enablement</th>
<th>Type</th>
<th>Object Id: Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alternate</td>
<td>Index</td>
<td>13: Comments (PUB.Customer)</td>
</tr>
<tr>
<td>Alternate</td>
<td>Index</td>
<td>14: Country-Post (PUB.Customer)</td>
</tr>
<tr>
<td>Alternate</td>
<td>Index</td>
<td>15: Name (PUB.Customer)</td>
</tr>
</tbody>
</table>

### Area 11: "Order Index Area" - Primary Buffer Pool

<table>
<thead>
<tr>
<th>Object Enablement</th>
<th>Type</th>
<th>Object Id: Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Index</td>
<td>22: Order-Date (PUB.Order)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>23: Sales-Rep (PUB.Order)</td>
</tr>
<tr>
<td>Default</td>
<td>Index</td>
<td>25: item-num (PUB.Order-Line)</td>
</tr>
</tbody>
</table>
**Notes**

- VIEWB2 maybe run online or offline.
- 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 the “PROUTIL ENABLEB2” section on page 21–49 or the Data Dictionary.
- For more information on removing objects from the alternate buffer pool, see the “PROUTIL DISABLEB2” section on page 21–36 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

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proutil -C wbreak-compiler src-file rule-num</td>
<td></td>
</tr>
</tbody>
</table>

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.
This chapter describes the OpenEdge database administration utility PROSTRCT.
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 ]] [-Passphrase]
```

Parameters

```
qualifier
```

Specifies the qualifier that you want to use. You can supply the following qualifiers:

- ADD
- ADDONLINE
- BUILDB
- 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 ]

Specifies the userid and password of an authenticated database administrator.

-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 Chapter 10, “Transparent Data Encryption,” 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 Chapter 1, “Creating and Deleting Databases,” 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 Chapter 19, “Database Startup Parameters,” 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 the “Validating structure description files” section on page 15–22.

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 the “PROSTRCT ADDONLINE qualifier” section on page 22–5.

- The new structure description file cannot identify existing extent files. It can only contain the definitions for new extent files. See Chapter 1, “Creating and Deleting Databases,” 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 the “Validating structure description files” section on page 15–22.

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)
```

- If ADDONLINE fails, the newly created files will be deleted.
- For a discussion and examples of PROSTRCT ADDONLINE, see the “OpenEdge Structure Add Online utility” section on page 15–11.
PROSTRCT BUILDB qualifier

Re-creates a control area from the structure description (.st) of an existing database.

```
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:** If 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 BUILDB will not be accurate.

**Notes**

- BUILDB does only minimal validation of the resulting control area.
- If your database is enabled for Replication, it is disabled after running BUILDB.
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 the “Disabling a cluster-enabled database” section on page 12-12.

- 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.

```
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 Chapter 2, “OpenEdge RDBMS Limits,” 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 the “Creating a structure description file” section on page 1–3.

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 .bn extension, where `n` represents the number of extents created.
For AI extents, the filename is the database name with a .an 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 .tn extension.

For Schema area extents, the filename is the database name with a .dn 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 .dn 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 Chapter 15, “Maintaining Database Structure,” for more information about the CREATE qualifier with the PROSTRCT utility.
**PROSTRCT LIST qualifier**

Creates a structure description (.st) file for an OpenEdge database.

```
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.

It 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 Chapter 15, “Maintaining Database Structure,” 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`

Indicates 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`

Indicates 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 Chapter 15, “Maintaining Database Structure,” for more information about the REMOTE 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 the “PROSTRCT LIST
  qualifier” section on page 22–11.

- See the “Managing after-imaging files” section on page 7–9 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 Utility

PROSTRCT STATISTICS qualifier

Displays statistics about storage information about the specified database.

Syntax

<table>
<thead>
<tr>
<th>prostrct statistics db-name</th>
</tr>
</thead>
</table>

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 Chapter 15, “Maintaining Database Structure,” 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 the “Restoring a database” section on page 5–20.
This chapter describes the OpenEdge database administration utility RFUTIL.
RFUTIL utility syntax

Performs various roll-forward recovery activities, depending on the qualifier you supply.

Syntax

```
rfutil db-name -C qualifier
    [-userid username [-password passwd ]]
    [-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 23–1.

*userid username [-password passwd ]*

Specifies the userid and password of an authenticated database administrator.

*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 Chapter 10, “Transparent Data Encryption,” for more information on transparent data 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:** RFUTIL and its qualifiers support the use of internationalization startup parameters such as *-cpinternal codepage* and *-cpstream codepage*. See Chapter 19, “Database Startup Parameters,” for a description of database-related internationalization startup parameters.
Table 23–1 lists the RFUTIL qualifiers. See the appropriate section for each of the qualifiers for more detailed information.

**Table 23–1: RFUTIL utility qualifiers**

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AIARCHIVER DISABLE</td>
<td>Disables the AI File Management Utility.</td>
</tr>
<tr>
<td>AIARCHIVER ENABLE</td>
<td>Enables the AI File Management Utility.</td>
</tr>
<tr>
<td>AIARCHIVER END</td>
<td>Shuts down the AI File Management Utility daemon.</td>
</tr>
<tr>
<td>AIARCHIVER SETDIR</td>
<td>Changes the archive destination directory list for the AI File Management Utility daemon.</td>
</tr>
<tr>
<td>AIARCHIVER SETINTERVAL</td>
<td>Changes the value of the archive interval for the AI File Management daemon.</td>
</tr>
<tr>
<td>AIARCHIVE EXTENT</td>
<td>Archives the specified AI extent when AI File Management is not enabled.</td>
</tr>
<tr>
<td>AIMAGE AIOFF</td>
<td>Disables after-imaging for a database during maintenance.</td>
</tr>
<tr>
<td>AIMAGE BEGIN</td>
<td>Enables after-imaging for a database.</td>
</tr>
<tr>
<td>AIMAGE END</td>
<td>Disables after-imaging for a database.</td>
</tr>
<tr>
<td>AIMAGE EXTENT EMPTY</td>
<td>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.</td>
</tr>
<tr>
<td>AIMAGE EXTENT FULL</td>
<td>Displays the pathname of the oldest filled file.</td>
</tr>
<tr>
<td>AIMAGE EXTENT LIST</td>
<td>Displays information about extents.</td>
</tr>
<tr>
<td>AIMAGE EXTRACT</td>
<td>Extracts the active blocks of an AI extent and writes them to a file.</td>
</tr>
<tr>
<td>AIMAGE NEW</td>
<td>Switches after-imaging to the next AI extent.</td>
</tr>
<tr>
<td>AIMAGE QUERY</td>
<td>Queries an AI extent for status.</td>
</tr>
<tr>
<td>AIMAGE SCAN</td>
<td>Scans AI files and displays information from that file.</td>
</tr>
<tr>
<td>AIMAGE TRUNCATE</td>
<td>Truncates all of the variable-length AI extents and optionally sets the AI block size with the After-image Block Size (-aiblocksize) parameter.</td>
</tr>
<tr>
<td>AIVERIFY PARTIAL</td>
<td>Verifies the AI extent by validating AI files before they are applied. This verification focuses on the AI block and the r1note structure (note headers), while providing additional information in error messages that specify issues found in the AI extent.</td>
</tr>
<tr>
<td>AIVERIFY FULL</td>
<td>Verifies the full AI extent, including verification of note data members.</td>
</tr>
</tbody>
</table>
Table 23–1: RFUTIL utility qualifiers (2 of 2)

<table>
<thead>
<tr>
<th>Qualifier</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MARK BACKEDUP</td>
<td>Marks the database file, indicating that you have just completed a backup of the database.</td>
</tr>
<tr>
<td>ROLL FORWARD</td>
<td>Reconstructs a database by applying to that database all notes stored in the AI file.</td>
</tr>
<tr>
<td>ROLL FORWARD OPLOCK</td>
<td>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.</td>
</tr>
<tr>
<td>ROLL FORWARD RETRY</td>
<td>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.</td>
</tr>
<tr>
<td>ROLL OPUNLOCK</td>
<td>Cancels the roll forward protection established by ROLL FORWARD OPLOCK.</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Updates the after-image sequence number of a database.</td>
</tr>
</tbody>
</table>

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 the “AI File Management utility” section on page 7–16 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 Chapter 7, “After-imaging,” for 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.

Note

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.

**Note**

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 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 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 the “PROBKUP utility” section on page 24–13 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.
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. 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 the “PROBKUP utility” section on page 24–13, for enabling after-imaging when the database is online.
  - See the “RFUTIL AIMAGE BEGIN qualifier” section on page 23–12, 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.

Note

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

Note

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 the “Managing after-imaging files” section on page 7–9 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

```
rutil 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-option search-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 Table 23–2.

Table 23–2: AIMAGE QUERY options

<table>
<thead>
<tr>
<th>Query option</th>
<th>Value returned</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTNUM</td>
<td>The extent number</td>
</tr>
<tr>
<td>STATUS</td>
<td>The status of the extent: FULL, LOCKED, BUSY, EMPTY</td>
</tr>
<tr>
<td>TYPE</td>
<td>The type of file: Fixed Length or Variable Length</td>
</tr>
<tr>
<td>SIZE</td>
<td>The size of the extent in 1K blocks</td>
</tr>
<tr>
<td>USED</td>
<td>The amount of the extent used in 1K blocks</td>
</tr>
<tr>
<td>NAME</td>
<td>The full filename of the extent</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>The sequence number</td>
</tr>
<tr>
<td>STARTDATE</td>
<td>The date and time after-imaging started writing to the extent</td>
</tr>
<tr>
<td>ALL</td>
<td>All of the above information</td>
</tr>
</tbody>
</table>

$search-option$

Specifies how you are identifying the AI extent to query. Supply one of the options from column one of Table 23–3.
**search-value**

Specifies the match criteria for the *search-option*. Supply the value indicated in column two of Table 23–3 that matches your *search-option*.

**Table 23–3: AIMAGE QUERY search options and values**

<table>
<thead>
<tr>
<th>Search-option</th>
<th>Search-value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTNUM</td>
<td>Integer value, in a file named, dbname.an, the EXTNUM is the value n</td>
</tr>
<tr>
<td>NAME</td>
<td>Character string, the extent file name</td>
</tr>
<tr>
<td>SEQUENCE</td>
<td>Integer value, the sequence number of the extent</td>
</tr>
</tbody>
</table>

**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
```

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.

Notes

- The AIMAGE SCAN qualifier fails if:
  - You omit the after-image area (-a) 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 AI extents 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 Chapter 14, “Managing Performance,” 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 r1note structure (note headers), while providing additional information in error messages that specify issues found in the AI extent.

Syntax

```shell
rfutil db-name -C aiverify partial-a ai-name
```

Parameters

- `db-name`
  
  Specifies the database you are using.
  
- `-a ai-name`
  
  Identifies the AI extent name.
RFUTIL Utility

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 r1note structure (note headers), this qualifier verifies note data members.

Syntax

```
rutil db-name -C aiverify full-a ai-name
```

Parameters

- `db-name`
  
  Specifies the database you are using.
  
- `-a ai-name`
  
  Identifies the AI extent name.
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
```

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.

- `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) 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.

```
rfutil db-name -C roll forward oplock -a ai-name
```

**Parameters**

*db-name*

Specifies the database you are using.

*ai-name*

Identifies the AI name of the specified database.

**Note**

All qualifiers accepted by ROLL FORWARD are also accepted by ROLL FORWARD OPLOCK. See the “RFUTIL ROLL FORWARD qualifier” section on page 23–26 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-area
```

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.
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.

**Note**

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

$db-name$

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 the “Roll forward with the OPLOCK qualifier” section on page 7–31 for more information.
RFUTIL Utility

RFUTIL SEQUENCE qualifier

Updates the database after-image sequence number.

Syntax

```
rfutil db-name -C sequence
```

*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.

**Note**

RFUTIL SEQUENCE is only needed when an AI extent switch occurs after the source database was backed up, and before a copy is made.
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, specifically:

- Parameters for authentication
- DBMAN utility
- DBTOOL utility
- PROADSV utility
- PROBKUP utility
- PROCLUSTER utility
- PROCOPY utility
- PRODB utility
- PRODEL utility
- Progress Explorer 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 ]**

  Specifies the userid and password of an authenticated database administrator.

- **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 Chapter 10, “Transparent Data Encryption,” 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.
DBMAN utility

Starts, stops, or queries a database. Before you can use the DBMAN command-line utility, you must use the OpenEdge Explorer or Progress Explorer configuration tool 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 the OpenEdge Explorer or Progress Explorer configuration tool.

- DBMAN supports the use of internationalization startup parameters such as, `-cpinternal codepage` and `-cpstream codepage`. See Chapter 19, “Database Startup Parameters,” for a description of each database-related internationalization startup parameter.

- The `conmgr.properties` file stores the database, configuration, and server group properties. For example:

```
# # Connection Manager Properties File
# # version 1.1
# % Oct 31, 2005 1:59:37 PM
# # 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          # -aiBuffers
# afterimageprocess=false      # n/a
# afterimagedelaywrites=3      # MF
# # asynchronouspagewriters=1  # n/a
# beforeimagebuffers=5         # -bibufs
# beforeimageclusterage=60     # \G
# beforeimagelock=60           # -G
# beforeimageprocess=true      # n/a
# # blocksindatablock=0         # -B (calculated as 8*(n))
# casetablename=basic         # -cpcase
# collationtable=basic         # -cpcoll
# conversionmap=convmap.cp     # -convmap
# crashprotection=true         # -i
# # databasecodecpage=basic    # -cpdb
# directio=false               # -directio
# hashtableentries=0           # -hash (calculated as (-B)/4)
```

(1 of 3)
# internalcodepage=iso8859-1     # -cpinternal
# locktableentries=10000        # -L
# logcharacterset=iso8859-1     # -cplog
# maxservers=4                 # -Mn
# maxusers=20                  # -n
# nap=1                       # -nap
# napmax=1                    # -napmax
# pagewritermaxbuffers=25      # -pwmax
# pagewriterqueuedelay=100     # -pwendelay
# pagewriterqueue=1           # -pwqmin
# pagewriterscan=1            # -pwsscan
# 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 Figure 24–1.

```
DATABASE TOOLS MENU - 10.1c
--------------------------
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 Block(s)
6. Record Fixup
7. Schema Validation
9. Enable/Disable File Logging
Q. Quit
Choice:
```

Figure 24–1: DBTOOL main menu

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`

In the generated report, three asterisks (*** in the Error column indicate SQL width or date violations.

- **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. 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.

- **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. 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.

• **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.

For the first six 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 1 through 7 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)?
     ```

   - **Area** — Enter the area number at the following prompt:

     ```
     <area>: (Area number or all)?
     ```

   - **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 0-3)?
     ```
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.
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 7832.

- **-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 7832
```

- 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, ODBC DataServer or ORACLE DataServer, AppServer, NameServer, WebSpeed® Transaction Server, or WebSpeed Messenger. The AdminServer must be running in order to use any of the Progress Explorer configuration tools or command-line configuration utilities, such as DBMAN.


- PROADSV supports the use of internationalization startup parameters such as `-cpinternal codepage` and `-cpstream codepage`. See Chapter 19, “Database Startup Parameters,” 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 ] ]
   [ -estimate | -vs n | -bf n | -verbose | -scan
     | -10 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 or a floppy diskette. For Windows, use `\.	ape0` 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 Chapter 7, “After-imaging,” 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 the “PROUTIL DESCRIBE qualifier” section on page 21–30 for more information. See Chapter 10, “Transparent Data Encryption,” 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 the “AI File Management utility” section on page 7–16.

**-aiarcdircreate**

Directs the AI File Management utility to create the directories specified by `-aiarchdir`.

**-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.

**-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 Chapter 5, “Backing Up a Database,” 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 Chapter 7, “After-imaging,” 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 the “Incremental backups” section on page 4–4 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 Chapter 5, “Backing Up a Database,” for more information about performing backups.

PROBKUP supports the use of internationalization startup parameters such as -cpi internal codepage and -cpstream codepage. See Chapter 19, “Database Startup Parameters,” for a description of each database-related internationalization startup parameter.
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 | isalive |
```

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

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 Chapter 7, “After-imaging,” 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
```

isalive

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
```

Note

- 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.
PROCOPY utility

Copies an existing database.

Syntax

```
procopy source-db-name target-db-name [-newinstance] [-silent]
```

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.

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 Chapter 19, “Database Startup Parameters,” for a description of each database-related internationalization startup parameter.

- Databases from releases prior to 10.1A do not have the schema support for a database GUID. When copying one of these older databases, the GUID field will not be added to the database. If the -newinstance qualifier is used, it is silently ignored. Use PROUTIL UPDATESCHEMA to determine if your schema is up to date. For more information, see the “PROUTIL UPDATESCHEMA qualifier” section on page 21–107.

- 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 Chapter 2, “OpenEdge RDBMS Limits,” 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 from releases prior to 10.1A do not have the schema support for a database
  GUID. When copying one of these older databases, the GUID field will not be added to
  the database. If the -newinstance qualifier is used, it is silently ignored. Use PROUTIL
  UPDATESCHEMA to determine if your schema is up to date. For more information, see
  the “PROUTIL UPDATESCHEMA qualifier” section on page 21–107.

• 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 Chapter 19, “Database Startup Parameters,” 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 Chapter 19, “Database Startup Parameters,” 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.
Progress Explorer utility

The Progress Explorer utility is part of a system administration framework that provides a consistent interface for managing all OpenEdge products installed on your network. Within this context, Progress Explorer provides configuration tools by means of a graphical user interface (GUI) for the administration of your OpenEdge database servers. For example, you can use Progress Explorer to perform the following tasks on a local or remote host:

- Connect to an AdminServer
- Start and stop OpenEdge databases
- Configure database properties through a Property Editor window

For instructions on using Progress Explorer to complete these tasks, see the Progress Explorer online help.

To launch Progress Explorer from the Start menu, choose Programs→OpenEdge→Progress Explorer Tool.

Notes

- 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 the Progress Explorer configuration tools or command-line configuration utilities to manage your database. In Windows-based systems, the AdminServer starts automatically and runs as a Windows 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.

- The database configurations you create with Progress Explorer are saved in the conmgr.properties file. Do not edit the conmgr.properties file directly.

- Progress Explorer is a snap-in to the Microsoft Management Console (MMC), the system and network administration tools framework. For more information about MMC, see the MMC online help.
PROLOG utility

Truncates the log file.

Syntax

```bash
prolog db-name [ -online ]
```

Parameters

- `db-name`
  Specifies the database log you want to truncate. Do not include the `.db` suffix.
- `online`
  Specifies that the database whose log is being truncated is currently online.

Notes

- If you want to save the log file, use operating system utilities to back it up before using PROLOG.
- See Chapter 17, “Logged Data,” for more information about log files.
- PROLOG supports the use of internationalization startup parameters such as `-cpi internal codepage` and `-cpi stream codepage`. See Chapter 19, “Database Startup Parameters,” 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-name device-name { -vp | -vf | -list } [-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 Chapter 5, “Backing Up a Database,” 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.

`-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

- 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.
- 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.
Figure 24–2 shows an example of the output generated by the -list qualifier.

<table>
<thead>
<tr>
<th>Area Name</th>
<th>Size</th>
<th>Records/Block</th>
<th>Area Number</th>
<th>Cluster Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schema Area</td>
<td>11264</td>
<td>32</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Info Area</td>
<td>1024</td>
<td>32</td>
<td>7</td>
<td>1</td>
</tr>
<tr>
<td>Customer/Order Area</td>
<td>6656</td>
<td>32</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Primary Index Area</td>
<td>112</td>
<td>32</td>
<td>9</td>
<td>8</td>
</tr>
<tr>
<td>Customer Index Area</td>
<td>256</td>
<td>32</td>
<td>10</td>
<td>64</td>
</tr>
<tr>
<td>Order Index Area</td>
<td>8192</td>
<td>32</td>
<td>11</td>
<td>64</td>
</tr>
<tr>
<td>Encryption Policy Area</td>
<td>20448</td>
<td>32</td>
<td>12</td>
<td>64</td>
</tr>
<tr>
<td>Audit Area</td>
<td>4608</td>
<td>32</td>
<td>20</td>
<td>8</td>
</tr>
<tr>
<td>Audit Index</td>
<td>8704</td>
<td>32</td>
<td>22</td>
<td>8</td>
</tr>
</tbody>
</table>

**Figure 24–2: Sample PROREST -list output**

Use the output of PROREST -list to calculate the size of each restored area as follows:

\[
\text{area-size} = \left( \frac{\text{Size}}{\text{records-per-block}} \right) \times \text{database-block-size}
\]

For example, the size of the restored schema area is:

\[
\text{area-size} = \left( \frac{11264}{32} \right) \times 4096 = 1,441,792
\]

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 Chapter 19, “Database Startup Parameters,” for a description of each database-related internationalization startup parameter.
This chapter describes the OpenEdge database administration utilities for SQL. The utilities include:

- 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,...
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.

- **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 utility

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
Examples

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
```

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 Character 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:

```
sql dump -t ""Yearly Profits"""" -u xxx -a yyy db_name
```

- 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

```sql
sqlload -u user_name [ -a password ] [ -C code_page_name ]
-t [ owner_name. ]table_name1 [ ,owner_name. ]table_name2, ...]
[ -1 log_file_name ] [ -b badfile_name ] [ -e max_errors ]
[ -F comma | quote ] [ -c commit_frequency ]
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.
SQLLOAD utility

-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.

-n

Directs SQLLOAD to check for syntax errors without loading any rows.

-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.

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.

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.
Examples

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
```

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 ]
] ]
] ]
```

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.dfsq1

  Specifies the output file where SQLSCHEMA writes the definitions. When specified, the file extension name must be .dfsq1. If output_file_name is omitted, SQLSCHEMA writes the definitions to the screen.

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.

Examples

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
```

**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.
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.

This chapter contains the following sections:

- Update access to virtual system tables
- Accessing VSTs with ABL or SQL
- Virtual system table summaries
- OpenEdge virtual system table schema descriptions
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:

```
proutil db-name -C updatevst
```

`db-name`

Specifies the database you are using.
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:

**ABL query**

```abl
FOR EACH _database-Feature:
   IF _DBFeature_Enabled = "1" THEN
   DISPLAY _DBFeature_Name.
```

**SQL query**

```sql
SELECT
   "_DBFeature_name" FROM pub."_database-Feature"
   WHERE "_DBFeature_Enabled" = 1;
```

The result of running either query against the sports database is:

<table>
<thead>
<tr>
<th>_DBFeature_Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>64 Bit DBKEYS</td>
</tr>
<tr>
<td>Large Keys</td>
</tr>
<tr>
<td>64 Bit Sequences</td>
</tr>
</tbody>
</table>

See the sections that follow for descriptions of the Virtual System Tables and their fields.
Virtual system table summaries

This section summarizes the OpenEdge RDBMS’s virtual system tables. Table 26–1 briefly describes each table and refers you to a more detailed schema description for each table.

<table>
<thead>
<tr>
<th>Virtual system table</th>
<th>Description</th>
<th>Where to find in this chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>After-image log activity file (_ActAILog)</td>
<td>Displays after-image log activity, such as the number of after-image writes, records and bytes written, busy buffer waits, and log force waits.</td>
<td>Table 26–2</td>
</tr>
<tr>
<td>Before-image log activity file (_ActBILog)</td>
<td>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.</td>
<td>Table 26–3</td>
</tr>
<tr>
<td>Buffer activity file (_ActBuffer)</td>
<td>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.</td>
<td>Table 26–4</td>
</tr>
<tr>
<td>Index activity file (_ActIndex)</td>
<td>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.</td>
<td>Table 26–5</td>
</tr>
<tr>
<td>Input/Output activity file (_ActIOFile)</td>
<td>Displays information about input/output activity, including the number of reads, writes, and extends for each file.</td>
<td>Table 26–6</td>
</tr>
<tr>
<td>Input/Output type activity file (_ActIOType)</td>
<td>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.</td>
<td>Table 26–7</td>
</tr>
<tr>
<td>Lock table activity file (_ActLock)</td>
<td>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.</td>
<td>Table 26–8</td>
</tr>
<tr>
<td>Other activity file (_ActOther)</td>
<td>Displays information about miscellaneous activity, including the number of commits, undo operations, semaphore waits, master block flushes, and database up time.</td>
<td>Table 26–9</td>
</tr>
</tbody>
</table>
Table 26–1: OpenEdge virtual system tables

<table>
<thead>
<tr>
<th>Virtual system table</th>
<th>Description</th>
<th>Where to find in this chapter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page writer activity file (ActPWs)</td>
<td>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.</td>
<td>Table 26–10</td>
</tr>
<tr>
<td>Record activity file (ActRecord)</td>
<td>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.</td>
<td>Table 26–11</td>
</tr>
<tr>
<td>Server activity file (ActServer)</td>
<td>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.</td>
<td>Table 26–12</td>
</tr>
<tr>
<td>Space allocation activity file (ActSpace)</td>
<td>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.</td>
<td>Table 26–13</td>
</tr>
<tr>
<td>Summary activity file (ActSummary)</td>
<td>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.</td>
<td>Table 26–14</td>
</tr>
<tr>
<td>Area status file (AreaStatus)</td>
<td>Displays a variety of data about the status of areas.</td>
<td>Table 26–15</td>
</tr>
<tr>
<td>Virtual system table</td>
<td>Description</td>
<td>Where to find in this chapter</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>-------------------------------</td>
</tr>
<tr>
<td>Area threshold file</td>
<td>Displays information about the consumption of available address space in an area.</td>
<td>Table 26–16</td>
</tr>
<tr>
<td>(_AreaThreshold)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Block file (_Block)</td>
<td>Displays information about a specific block.</td>
<td>Table 26–17</td>
</tr>
<tr>
<td>Buffer status file</td>
<td>Displays 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.</td>
<td>Table 26–18</td>
</tr>
<tr>
<td>(_BuffStatus)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checkpoint file</td>
<td>Displays information about each checkpoint, including the checkpoint number and beginning time, the time required to complete the checkpoint, and the number of modified blocks scheduled to be written. The file also describes APW-written blocks.</td>
<td>Table 26–19</td>
</tr>
<tr>
<td>(_Checkpoint)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code features</td>
<td>Displays the list of database features that the code supports and requires of all databases that are opened with it.</td>
<td>Table 26–20</td>
</tr>
<tr>
<td>(_Code-Feature)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database connection</td>
<td>Displays information about two-phase commit, batch user, and device connections.</td>
<td>Table 26–21</td>
</tr>
<tr>
<td>file (_Connect)</td>
<td><strong>Note:</strong> If you are connecting to a database on UNIX through a Telnet session using <code>-S port -H localhost</code>, a TTY session ID is appended to the field.</td>
<td></td>
</tr>
<tr>
<td>Database features</td>
<td>Displays the list of features that are active and/or enabled within the database.</td>
<td>Table 26–22</td>
</tr>
<tr>
<td>file (_Database-Feature)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database status file</td>
<td>Displays a wide variety of status data.</td>
<td>Table 26–23</td>
</tr>
<tr>
<td>(_DbStatus)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Database file status</td>
<td>Displays the filename, file size, and the size of every database file and extent.</td>
<td>Table 26–24</td>
</tr>
<tr>
<td>file (_Filelist)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index statistics file</td>
<td>Displays statistics on the number of accesses to a specific range of indexes.</td>
<td>Table 26–25</td>
</tr>
<tr>
<td>(_IndexStat)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Latch statistics file</td>
<td>Displays statistics on latch and latch queue activity.</td>
<td>Table 26–26</td>
</tr>
<tr>
<td>(_Latch)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>License management</td>
<td>Provides information about the number of users and connections.</td>
<td>Table 26–27</td>
</tr>
<tr>
<td>(_License)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Lock table status file</td>
<td>Displays the status of the lock table, including the user number, the user name, lock type, RECID number, flags, and chain.</td>
<td>Table 26–28</td>
</tr>
<tr>
<td>(_Lock)</td>
<td><strong>Note:</strong> Use caution when querying. _Lock. Doing so consumes large amounts of system resources.</td>
<td></td>
</tr>
<tr>
<td>Virtual system table</td>
<td>Description</td>
<td>Where to find in this chapter</td>
</tr>
<tr>
<td>--------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------</td>
</tr>
<tr>
<td>Lock request file (_LockReq)</td>
<td>Displays information about lock requests, including user name and number, record locks and waits, schema locks and waits, and transaction locks and waits.</td>
<td>Table 26–29</td>
</tr>
<tr>
<td>Logging file (_Logging)</td>
<td>Includes before-image and after-image logging information and two-phase commit status.</td>
<td>Table 26–30</td>
</tr>
<tr>
<td>Master block file (_MstrBlk)</td>
<td>Displays before-image and after-image information about the master block and other master block status.</td>
<td>Table 26–31</td>
</tr>
<tr>
<td>User connection (_MyConnection)</td>
<td>Provides information about a user and the number of private read-only buffers allowed and in use.</td>
<td>Table 26–32</td>
</tr>
<tr>
<td>Resource queue statistics file (_Resrc)</td>
<td>Displays statistics on resource queue utilization.</td>
<td>Table 26–33</td>
</tr>
<tr>
<td>Segments file (_Segments)</td>
<td>Reports the segment number, segment size, and the number of free and used segments.</td>
<td>Table 26–34</td>
</tr>
<tr>
<td>Servers file (_Servers)</td>
<td>Displays status of OpenEdge servers running on the system, such as 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.</td>
<td>Table 26–35</td>
</tr>
<tr>
<td>Startup file (_Startup)</td>
<td>Displays the values of the database startup parameters that were used.</td>
<td>Table 26–36</td>
</tr>
<tr>
<td>Index and table statistics range (_StatBase)</td>
<td>Displays basic table and index statistics.</td>
<td>Table 26–37</td>
</tr>
<tr>
<td>Table statistics file (_TableStat)</td>
<td>Displays statistics on the number of accesses to a specific range of tables.</td>
<td>Table 26–38</td>
</tr>
<tr>
<td>Transaction file (_Trans)</td>
<td>Includes information such as transaction number, state, start time, duration, user number, coordinator name, and transaction.</td>
<td>Table 26–39</td>
</tr>
<tr>
<td>Transaction end lock statistics (_TxeLock)</td>
<td>Includes statistics about Transaction End Locks.</td>
<td>Table 26–40</td>
</tr>
<tr>
<td>User index activity (_UserIndexStat)</td>
<td>Displays statistics on the number of accesses to a specific range of indexes by user.</td>
<td>Table 26–41</td>
</tr>
<tr>
<td>Database input/output file (_UserIO)</td>
<td>Displays information about the database input/output operations, including user number and name and the number of accesses, reads, and writes.</td>
<td>Table 26–42</td>
</tr>
<tr>
<td>Virtual system table</td>
<td>Description</td>
<td>Where to find in this chapter</td>
</tr>
<tr>
<td>----------------------------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>------------------------------</td>
</tr>
<tr>
<td>Record locking table file (_UserLock)</td>
<td>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.</td>
<td>Table 26–43</td>
</tr>
<tr>
<td>User status (_UserStatus)</td>
<td>Displays the progress of PROUTIL and PROBKUP utilities.</td>
<td>Table 26–44</td>
</tr>
<tr>
<td>User table activity (_UserTableStat)</td>
<td>Displays statistics on the number of accesses to a specific range of tables by user.</td>
<td>Table 26–46</td>
</tr>
</tbody>
</table>
OpenEdge virtual system table schema descriptions

Table 26–2 through Table 26–45 describe the schema for each virtual system table.

**Table 26–2: After-image log activity file (_ActAILog)**

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_AiLog-AIWWrites</td>
<td>INT64</td>
<td>Number of after-image writes performed by the after-image writer (AIW). This is a subset of the total number of AI writes.</td>
</tr>
<tr>
<td>_AiLog-BBuffWaits</td>
<td>INT64</td>
<td>Number of busy buffer waits.</td>
</tr>
<tr>
<td>_AiLog-BytesWritn</td>
<td>INT64</td>
<td>Amount of AI data written to the AI file, in bytes.</td>
</tr>
<tr>
<td>_AiLog-ForceWaits</td>
<td>INT64</td>
<td>Number of waiting-for-commit records to be written to disk.</td>
</tr>
<tr>
<td>_AiLog-NoBufAvail</td>
<td>INT64</td>
<td>Total number of times a process had to wait because a buffer was not available.</td>
</tr>
<tr>
<td>_AiLog-PartialWrt</td>
<td>INT64</td>
<td>Number of writes to the AI file made before the AI buffer is full.</td>
</tr>
<tr>
<td>_AiLog-RecWriten</td>
<td>INT64</td>
<td>Number of records written to the AI file.</td>
</tr>
<tr>
<td>_AiLog-TotWrites</td>
<td>INT64</td>
<td>Total number of writes to the AI file.</td>
</tr>
<tr>
<td>_AiLog-Trans</td>
<td>INT64</td>
<td>Number of transactions committed to the AI file.</td>
</tr>
<tr>
<td>_AiLog-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the AI file was open.</td>
</tr>
</tbody>
</table>

**Table 26–3: Before-image log activity file (_ActBiLog)** *(1 of 2)*

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_BiLog-BBuffWaits</td>
<td>INT64</td>
<td>Number of times a process had to wait for a buffer that was busy.</td>
</tr>
<tr>
<td>_BiLog-BIWWrites</td>
<td>INT64</td>
<td>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.</td>
</tr>
<tr>
<td>_BiLog-BytesRead</td>
<td>INT64</td>
<td>Number of bytes of data read from the BI file.</td>
</tr>
<tr>
<td>_BiLog-BytesWrttn</td>
<td>INT64</td>
<td>Number of bytes of data written to the BI file.</td>
</tr>
<tr>
<td>_BiLog-CIstrClose</td>
<td>INT64</td>
<td>Number of BI clusters filled and closed in preparation for reuse.</td>
</tr>
</tbody>
</table>
### Table 26–3: Before-image log activity file (_ActBILog) (2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_BiLog-EBuffWaits</td>
<td>INT64</td>
<td>Number of times a process had to wait because all buffers were full.</td>
</tr>
<tr>
<td>_BiLog-ForceWaits</td>
<td>INT64</td>
<td>Number of waiting-for-commit records to be written to disk.</td>
</tr>
<tr>
<td>_BiLog-ForceWrts</td>
<td>INT64</td>
<td>Number of waiting-for-commit records written to disk.</td>
</tr>
<tr>
<td>_BiLog-PartialWrts</td>
<td>INT64</td>
<td>Number of writes to the BI file made before the BI buffer is full.</td>
</tr>
<tr>
<td>_BiLog-RecRead</td>
<td>INT64</td>
<td>Number of BI records (notes) read from the BI file.</td>
</tr>
<tr>
<td>_BiLog-RecWritten</td>
<td>INT64</td>
<td>Number of BI records (notes) written to the BI file.</td>
</tr>
<tr>
<td>_BiLog-TotalWrts</td>
<td>INT64</td>
<td>Number of total writes to the BI file.</td>
</tr>
<tr>
<td>_BiLog-TotReads</td>
<td>INT64</td>
<td>Number of BI blocks read from the BI file to undo transactions.</td>
</tr>
<tr>
<td>_BiLog-Trans</td>
<td>INT64</td>
<td>Number of transactions committed to the BI file.</td>
</tr>
<tr>
<td>_BiLog-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the BI file was open.</td>
</tr>
</tbody>
</table>

### Table 26–4: Buffer activity file (_ActBuffer) (1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Buffer-APWEnq</td>
<td>INT64</td>
<td>Number of modified buffers placed on the APW queue for writing.</td>
</tr>
<tr>
<td>_Buffer-Chkpts</td>
<td>INT64</td>
<td>Number of checkpoint operations.</td>
</tr>
<tr>
<td>_Buffer-Deferred</td>
<td>INT64</td>
<td>Total number of changes to blocks that occurred before the blocks were written. Each deferred write is potentially an I/O operation saved.</td>
</tr>
<tr>
<td>_Buffer-Flushed</td>
<td>INT64</td>
<td>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.</td>
</tr>
<tr>
<td>_Buffer-LogicRds</td>
<td>INT64</td>
<td>Number of client requests for database block read operations.</td>
</tr>
<tr>
<td>_Buffer-LogicWrts</td>
<td>INT64</td>
<td>Number of client requests for database block write operations.</td>
</tr>
</tbody>
</table>
### Table 26–4: Buffer activity file (_ActBuffer)\(^1\)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Buffer-LRUSkips</td>
<td>INT64</td>
<td>Number of times a buffer on the LRU chain was skipped because it was locked or modified.</td>
</tr>
<tr>
<td>_Buffer-LRUwrs</td>
<td>INT64</td>
<td>Number of blocks written to free a buffer for a read operation.</td>
</tr>
<tr>
<td>_Buffer-Marked</td>
<td>INT64</td>
<td>Number of blocks scheduled to be written before the end of a checkpoint.</td>
</tr>
<tr>
<td>_Buffer-OSRds</td>
<td>INT64</td>
<td>Number of database blocks read from disk.</td>
</tr>
<tr>
<td>_Buffer-OSWrt</td>
<td>INT64</td>
<td>Number of database block writes to disk.</td>
</tr>
<tr>
<td>_Buffer-Trans</td>
<td>INT64</td>
<td>Number of transactions committed.</td>
</tr>
<tr>
<td>_Buffer-Uptime</td>
<td>INTEGER</td>
<td>Number of seconds the database was opened.</td>
</tr>
</tbody>
</table>

1. 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. The fields of the table are unchanged. Because the first _ActBuffer record contains total information, applications written prior to Release 10.2B to access the _ActBuffer table are unaffected by the additional records.

### Table 26–5: Index activity file (_ActIndex)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Index-Create</td>
<td>INT64</td>
<td>Number of new index entries generated</td>
</tr>
<tr>
<td>_Index-Delete</td>
<td>INT64</td>
<td>Number of index entries deleted</td>
</tr>
<tr>
<td>_Index-Find</td>
<td>INT64</td>
<td>Number of times an index entry was looked up</td>
</tr>
<tr>
<td>_Index-Free</td>
<td>INT64</td>
<td>Number of index blocks freed during deletes</td>
</tr>
<tr>
<td>_Index-Remove</td>
<td>INT64</td>
<td>Number of locks released at transaction end</td>
</tr>
<tr>
<td>_Index-Splits</td>
<td>INT64</td>
<td>Number of block splits resulting from index additions</td>
</tr>
<tr>
<td>_Index-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_Index-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
</tbody>
</table>
### Table 26–6: Input/output activity file (_ActIOFile)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IOFile-BufReads</td>
<td>INT64</td>
<td>Number of buffered read operations performed on the file</td>
</tr>
<tr>
<td>_IOFile-BufWrites</td>
<td>INT64</td>
<td>Number of buffered write operations performed on the file</td>
</tr>
<tr>
<td>_IOFile-Extends</td>
<td>INT64</td>
<td>Number of extend operations performed on the file</td>
</tr>
<tr>
<td>_IOFile-FileName</td>
<td>CHARACTER</td>
<td>Name of the file upon which operations are performed</td>
</tr>
<tr>
<td>_IOFile-Reads</td>
<td>INT64</td>
<td>Number of read operations performed on the file</td>
</tr>
<tr>
<td>_IOFile-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_IOFile-UbufReads</td>
<td>INT64</td>
<td>Number of unbuffered read operations performed on the file</td>
</tr>
<tr>
<td>_IOFile-UbufWrites</td>
<td>INT64</td>
<td>Number of unbuffered write operations performed on the file</td>
</tr>
<tr>
<td>_IOFile-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
<tr>
<td>_IOFile-Writes</td>
<td>INT64</td>
<td>Number of write operations performed on the file</td>
</tr>
</tbody>
</table>

### Table 26–7: Input/output type activity file (_ActIOType)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IOType-AiRds</td>
<td>INT64</td>
<td>Number of AI read operations</td>
</tr>
<tr>
<td>_IOType-AiWrts</td>
<td>INT64</td>
<td>Number of AI write operations</td>
</tr>
<tr>
<td>_IOType-BiRds</td>
<td>INT64</td>
<td>Number of BI read operations</td>
</tr>
<tr>
<td>_IOType-BiWrts</td>
<td>INT64</td>
<td>Number of BI write operations</td>
</tr>
<tr>
<td>_IOType-DataReads</td>
<td>INT64</td>
<td>Number of reads of data blocks</td>
</tr>
<tr>
<td>_IOType-DataWrts</td>
<td>INT64</td>
<td>Number of writes to data blocks</td>
</tr>
<tr>
<td>_IOType-IdxRds</td>
<td>INT64</td>
<td>Number of reads of index blocks</td>
</tr>
<tr>
<td>_IOType-IdxWrts</td>
<td>INT64</td>
<td>Number of writes to index blocks</td>
</tr>
<tr>
<td>_IOType-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_IOType-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
<tr>
<td>Field name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------</td>
<td>-----------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>_Lock-CancReq</td>
<td>INTEGER</td>
<td>Number of lock requests that were cancelled</td>
</tr>
<tr>
<td>_Lock-Downgrade</td>
<td>INTEGER</td>
<td>Number of locks that were downgraded</td>
</tr>
<tr>
<td>_Lock-ExclFind</td>
<td>INT64</td>
<td>Number of exclusive find requests granted</td>
</tr>
<tr>
<td>_Lock-ExclLock</td>
<td>INTEGER</td>
<td>Number of exclusive lock requests</td>
</tr>
<tr>
<td>_Lock-ExclReq</td>
<td>INTEGER</td>
<td>Number of requests for exclusive locks granted</td>
</tr>
<tr>
<td>_Lock-ExclWait</td>
<td>INTEGER</td>
<td>Number of times processes waited for an exclusive lock</td>
</tr>
<tr>
<td>_Lock-RecGetLock</td>
<td>INTEGER</td>
<td>Number of requests for record get locks granted</td>
</tr>
<tr>
<td>_Lock-RecGetReq</td>
<td>INTEGER</td>
<td>Number of record get locks granted</td>
</tr>
<tr>
<td>_Lock-RecGetWait</td>
<td>INTEGER</td>
<td>Number of times processes waited for record gets</td>
</tr>
<tr>
<td>_Lock-RedReq</td>
<td>INTEGER</td>
<td>Number of duplicate requests granted</td>
</tr>
<tr>
<td>_Lock-ShrFind</td>
<td>INT64</td>
<td>Number of share find requests granted</td>
</tr>
<tr>
<td>_Lock-ShrLock</td>
<td>INTEGER</td>
<td>Number of share lock requests granted</td>
</tr>
<tr>
<td>_Lock-ShrReq</td>
<td>INTEGER</td>
<td>Number of requests for share locks</td>
</tr>
<tr>
<td>_Lock-ShrWait</td>
<td>INTEGER</td>
<td>Number of times processes waited for share lock</td>
</tr>
<tr>
<td>_Lock-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_Lock-UpgLock</td>
<td>INTEGER</td>
<td>Number of requests for lock upgrades granted</td>
</tr>
<tr>
<td>_Lock-UpgReq</td>
<td>INTEGER</td>
<td>Number of requests to upgrade a lock from shared lock to exclusive lock</td>
</tr>
<tr>
<td>_Lock-UpgWait</td>
<td>INTEGER</td>
<td>Number of times processes waited for upgrades</td>
</tr>
<tr>
<td>_Lock-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
</tbody>
</table>
Table 26–9: Other activity file (\_ActOther)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Other-Commit</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_Other-FlushMblk</td>
<td>INT64</td>
<td>Number of times the database master block was written to disk</td>
</tr>
<tr>
<td>_Other-Trans</td>
<td>INT64</td>
<td>Transactions committed</td>
</tr>
<tr>
<td>_Other-Undo</td>
<td>INT64</td>
<td>Number of transactions rolled back</td>
</tr>
<tr>
<td>_Other-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
<tr>
<td>_Other-Wait</td>
<td>INT64</td>
<td>Number of times a process had to wait for a resource</td>
</tr>
</tbody>
</table>

Table 26–10: Page writer activity file (\_ActPWs)

(1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_PW-ApwQWrites</td>
<td>INT64</td>
<td>Number of buffers written to clear the APW queue</td>
</tr>
<tr>
<td>_PW-BuffsScanned</td>
<td>INT64</td>
<td>Number of buffers scanned during each cycle</td>
</tr>
<tr>
<td>_PW-BufsCkp</td>
<td>INT64</td>
<td>Number of buffers checkpointed</td>
</tr>
<tr>
<td>_PW-Checkpoints</td>
<td>INT64</td>
<td>Number of checkpoints that have occurred</td>
</tr>
<tr>
<td>_PW-CkpQWrites</td>
<td>INT64</td>
<td>Number of buffers written from the checkpoint queue</td>
</tr>
<tr>
<td>_PW-DBWrites</td>
<td>INT64</td>
<td>Number of database write operations performed by the APW</td>
</tr>
<tr>
<td>_PW-Flushed</td>
<td>INT64</td>
<td>Number of blocks that were not written during the checkpoint and had to be written all at once at the end of the checkpoint</td>
</tr>
<tr>
<td>_PW-Marked</td>
<td>INT64</td>
<td>Number of buffers that were scheduled to be written before the end of the checkpoint</td>
</tr>
<tr>
<td>_PW-ScanCycles</td>
<td>INT64</td>
<td>Number of scan cycles. During a scan cycle, the APW scans a portion of the buffer pool to look for modified buffers</td>
</tr>
<tr>
<td>_PW-ScanWrites</td>
<td>INT64</td>
<td>Number of buffers written during the scan cycle</td>
</tr>
</tbody>
</table>
Table 26–10: Page writer activity file (_ActPWs)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_PW-TotDBWrites</td>
<td>INT64</td>
<td>Total number of database write operations performed by all processes</td>
</tr>
<tr>
<td>_PW-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_PW-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
</tbody>
</table>

Table 26–11: Record activity file (_ActRecord)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Record-BytesCreat</td>
<td>INT64</td>
<td>Number of bytes created</td>
</tr>
<tr>
<td>_Record-BytesDel</td>
<td>INT64</td>
<td>Number of bytes deleted</td>
</tr>
<tr>
<td>_Record-BytesRead</td>
<td>INT64</td>
<td>Number of bytes read</td>
</tr>
<tr>
<td>_Record-BytesUpd</td>
<td>INT64</td>
<td>Number of bytes updated</td>
</tr>
<tr>
<td>_Record-FragCreat</td>
<td>INT64</td>
<td>Number of fragments created</td>
</tr>
<tr>
<td>_Record-FragDel</td>
<td>INT64</td>
<td>Number of fragments deleted</td>
</tr>
<tr>
<td>_Record-FragRead</td>
<td>INT64</td>
<td>Number of fragments read</td>
</tr>
<tr>
<td>_Record-FragUpd</td>
<td>INT64</td>
<td>Number of fragments updated</td>
</tr>
<tr>
<td>_Record-RecCreat</td>
<td>INT64</td>
<td>Number of records created</td>
</tr>
<tr>
<td>_Record-RecDel</td>
<td>INT64</td>
<td>Number of records deleted</td>
</tr>
<tr>
<td>_Record-RecRead</td>
<td>INT64</td>
<td>Number of records read</td>
</tr>
<tr>
<td>_Record-RecUpd</td>
<td>INT64</td>
<td>Number of records updated</td>
</tr>
<tr>
<td>_Record-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_Record-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
</tbody>
</table>

Table 26–12: Server activity file (_ActServer)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Server-ByteRec</td>
<td>INT64</td>
<td>Number of bytes received by the server</td>
</tr>
<tr>
<td>_Server-ByteSent</td>
<td>INT64</td>
<td>Number of bytes sent by the server</td>
</tr>
<tr>
<td>_Server-MsgRec</td>
<td>INT64</td>
<td>Number of network packets received by the server</td>
</tr>
<tr>
<td>_Server-MsgSent</td>
<td>INT64</td>
<td>Number of network packets sent by the server</td>
</tr>
<tr>
<td>_Server-QryRec</td>
<td>INT64</td>
<td>Number of query requests received</td>
</tr>
</tbody>
</table>
### Table 26–12: Server activity file (_ActServer) (2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Server-RecRec</td>
<td>INT64</td>
<td>Number of records received</td>
</tr>
<tr>
<td>_Server-RecSent</td>
<td>INT64</td>
<td>Number of records sent</td>
</tr>
<tr>
<td>_Server-TimeSlice</td>
<td>INT64</td>
<td>Number of query time slice switches</td>
</tr>
<tr>
<td>_Server-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_Server-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
</tbody>
</table>

### Table 26–13: Space allocation activity file (_ActSpace) (1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Space-AllocNewRm</td>
<td>INT64</td>
<td>Number of times space was allocated for a record or record fragment</td>
</tr>
<tr>
<td>_Space-BackAdd</td>
<td>INT64</td>
<td>Number of blocks added to the back of the rm chain</td>
</tr>
<tr>
<td>_Space-BytesAlloc</td>
<td>INT64</td>
<td>Number of bytes allocated for record fragments</td>
</tr>
<tr>
<td>_Space-DbExd</td>
<td>INT64</td>
<td>Number of times the database was extended</td>
</tr>
<tr>
<td>_Space-Examined</td>
<td>INT64</td>
<td>Number of blocks examined in the rm chain while looking for space for a record fragment</td>
</tr>
<tr>
<td>_Space-FromFree</td>
<td>INT64</td>
<td>Number of times space was allocated from the free chain</td>
</tr>
<tr>
<td>_Space-FromRm</td>
<td>INT64</td>
<td>Number of times space was allocated from the rm chain</td>
</tr>
<tr>
<td>_Space-Front2Back</td>
<td>INT64</td>
<td>Number of blocks moved from the front to the back of the rm chain</td>
</tr>
<tr>
<td>_Space-FrontAdd</td>
<td>INT64</td>
<td>Number of blocks added to the front of the rm chain</td>
</tr>
<tr>
<td>_Space-Locked</td>
<td>INT64</td>
<td>Number of locked rm entries removed</td>
</tr>
<tr>
<td>_Space-Removed</td>
<td>INT64</td>
<td>Number of blocks removed from the rm chain</td>
</tr>
<tr>
<td>_Space-RetFree</td>
<td>INT64</td>
<td>Number of times a block was returned to the free chain</td>
</tr>
</tbody>
</table>
Table 26–13: Space allocation activity file (_ActSpace) (2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Space-TakeFree</td>
<td>INT64</td>
<td>Number of times a block was used from the free chain</td>
</tr>
<tr>
<td>_Space-Trans</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_Space-UpTime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
</tbody>
</table>

Table 26–14: Summary activity file (_ActSummary)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Summary-AiWrites</td>
<td>INT64</td>
<td>Number of AI blocks written to disk</td>
</tr>
<tr>
<td>_Summary-BiReads</td>
<td>INT64</td>
<td>Number of BI blocks read</td>
</tr>
<tr>
<td>_Summary-BiWrites</td>
<td>INT64</td>
<td>Number of BI blocks written to disk</td>
</tr>
<tr>
<td>_Summary-Chkpts</td>
<td>INT64</td>
<td>Number of checkpoints that have been performed</td>
</tr>
<tr>
<td>_Summary-Commits</td>
<td>INT64</td>
<td>Number of transactions all users have committed</td>
</tr>
<tr>
<td>_Summary-DbAccesses</td>
<td>INT64</td>
<td>Number of times users have waited for shared and exclusive locks on a database buffer</td>
</tr>
<tr>
<td>_Summary-DbReads</td>
<td>INT64</td>
<td>Number of database blocks read</td>
</tr>
<tr>
<td>_Summary-DbWrites</td>
<td>INT64</td>
<td>Number of database blocks written to disk</td>
</tr>
<tr>
<td>_Summary-Flushed</td>
<td>INT64</td>
<td>Number of database buffers that have been flushed to disk because they were not written by the time the checkpoint ended</td>
</tr>
<tr>
<td>_Summary-RecCreat</td>
<td>INT64</td>
<td>Number of records created</td>
</tr>
<tr>
<td>_Summary-RecDel</td>
<td>INT64</td>
<td>Number of records deleted</td>
</tr>
<tr>
<td>_Summary-RecLock</td>
<td>INT64</td>
<td>Number of record locks used</td>
</tr>
<tr>
<td>_Summary-RecReads</td>
<td>INT64</td>
<td>Number of records read</td>
</tr>
<tr>
<td>_Summary-RecUpd</td>
<td>INT64</td>
<td>Number of records updated</td>
</tr>
<tr>
<td>_Summary-RecWait</td>
<td>INT64</td>
<td>Number of times users have waited to access a locked record</td>
</tr>
<tr>
<td>_Summary-TransComm</td>
<td>INT64</td>
<td>Number of transactions committed</td>
</tr>
<tr>
<td>_Summary-Undos</td>
<td>INT64</td>
<td>Number of transactions rolled back</td>
</tr>
<tr>
<td>_Summary-Uptime</td>
<td>INTEGER</td>
<td>Number of seconds the database was up</td>
</tr>
</tbody>
</table>
### Table 26–15: Area status file (._AreaStatus)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_AreaStatus-Areaname</td>
<td>CHARACTER</td>
<td>Area name</td>
</tr>
<tr>
<td>_AreaStatus-Areanum</td>
<td>INTEGER</td>
<td>Area number</td>
</tr>
<tr>
<td>_AreaStatus-Extents</td>
<td>INTEGER</td>
<td>Number of extents in an area</td>
</tr>
<tr>
<td>_AreaStatus-Freenum</td>
<td>INT64</td>
<td>Number of blocks on the free chain</td>
</tr>
<tr>
<td>_AreaStatus-Hiwater</td>
<td>INT64</td>
<td>Highest block used</td>
</tr>
<tr>
<td>_AreaStatus-Lastextent</td>
<td>CHARACTER</td>
<td>Name of extent where high-water mark resides</td>
</tr>
<tr>
<td>_AreaStatus-Rmnum</td>
<td>INT64</td>
<td>Number of blocks on the rm chain</td>
</tr>
<tr>
<td>_AreaStatus-Totblocks</td>
<td>INT64</td>
<td>Total number of blocks in an area</td>
</tr>
</tbody>
</table>

### Table 26–16: Area threshold file (._AreaThreshold)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_AreaThreshold</td>
<td>INTEGER</td>
<td>Number indicating how much of the addressable space in an area is consumed; possible values are:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 0 — Area is below 80 percent of maximum addressable threshold value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 — Area is above 80 percent of maximum addressable threshold value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2 — Area is above 90 percent of maximum addressable threshold value.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 4 — Area has reached the maximum addressable threshold and cannot be further extended.</td>
</tr>
<tr>
<td>_AreaThresholdArea</td>
<td>INTEGER</td>
<td>Area number</td>
</tr>
</tbody>
</table>

### Table 26–17: Block file (._Block)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Block-Area</td>
<td>INTEGER</td>
<td>Block area number</td>
</tr>
<tr>
<td>_Block-BkupCtr</td>
<td>INTEGER</td>
<td>Backup counter</td>
</tr>
<tr>
<td>_Block-Block</td>
<td>CHARACTER</td>
<td>Data section of the block</td>
</tr>
<tr>
<td>_Block-ChainType</td>
<td>CHARACTER</td>
<td>Chain type</td>
</tr>
</tbody>
</table>
## Table 26–17: Block file (_Block)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Block-Dbkey</td>
<td>INT64</td>
<td>Dbkey</td>
</tr>
<tr>
<td>_Block-NextDbkey</td>
<td>INT64</td>
<td>Next Dbkey in the appropriate chain</td>
</tr>
<tr>
<td>_Block-Type</td>
<td>CHARACTER</td>
<td>Type of block</td>
</tr>
<tr>
<td>_Block-Update</td>
<td>INTEGER</td>
<td>Number of times the block has been updated</td>
</tr>
</tbody>
</table>

## Table 26–18: Buffer status file (_BuffStatus)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_BfStatus-APWQ</td>
<td>INTEGER</td>
<td>Number of buffers on the page writer queue.</td>
</tr>
<tr>
<td>_BfStatus-CKPMarked</td>
<td>INT64</td>
<td>Number of buffers currently marked for checkpoint.</td>
</tr>
<tr>
<td>_BfStatus-CKPQ</td>
<td>INTEGER</td>
<td>Number of buffers on the checkpoint queue.</td>
</tr>
<tr>
<td>_BfStatus-HashSize</td>
<td>INTEGER</td>
<td>Size of the buffer hash table.</td>
</tr>
<tr>
<td>_BfStatus-LastCkpNum</td>
<td>INTEGER</td>
<td>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.</td>
</tr>
<tr>
<td>_BfStatus-LRU</td>
<td>INTEGER</td>
<td>Number of buffers on the least recently used (LRU) chain for both the primary and Alternate buffer pools.</td>
</tr>
<tr>
<td>_BfStatus-ModBuffs</td>
<td>INT64</td>
<td>Number of dirty (modified) buffers.</td>
</tr>
<tr>
<td>_BfStatus-TotBufs</td>
<td>INTEGER</td>
<td>Number of buffers in the buffer cache.</td>
</tr>
<tr>
<td>_BfStatus-UsedBufs</td>
<td>INTEGER</td>
<td>Number of buffers used.</td>
</tr>
</tbody>
</table>

## Table 26–19: Checkpoint File (_Checkpoint)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Checkpoint-ApwQ</td>
<td>INT64</td>
<td>Number of blocks written by the APW queue and replaced on the least recently used (LRU) chain by APWs</td>
</tr>
<tr>
<td>_Checkpoint-CptQ</td>
<td>INT64</td>
<td>Number of blocks written from the checkpoint queue by the APWs</td>
</tr>
<tr>
<td>_Checkpoint-Dirty</td>
<td>INT64</td>
<td>Number of modified blocks scheduled to be written</td>
</tr>
</tbody>
</table>
### Table 26–19: Checkpoint File (_Checkpoint)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Checkpoint-Flush</td>
<td>INT64</td>
<td>Total number of blocks not written during the checkpoint that had to be written all at once at the end of the checkpoint</td>
</tr>
<tr>
<td>_Checkpoint-Len</td>
<td>CHARACTER</td>
<td>Length of time required to complete the checkpoint</td>
</tr>
<tr>
<td>_Checkpoint-Scan</td>
<td>INT64</td>
<td>Number of blocks written by the APWs during the scan cycle</td>
</tr>
<tr>
<td>_Checkpoint-Time</td>
<td>CHARACTER</td>
<td>Time the checkpoint began</td>
</tr>
</tbody>
</table>

### Table 26–20: Code Features (_Code-Feature)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_CodeFeature_Name</td>
<td>CHARACTER</td>
<td>Name of the database feature.</td>
</tr>
<tr>
<td>_CodeFeature_Supported</td>
<td>CHARACTER</td>
<td>“1” indicates the code supports this database feature. “0” indicates the code does not support this feature.</td>
</tr>
<tr>
<td>_CodeFeature_Required</td>
<td>CHARACTER</td>
<td>“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.</td>
</tr>
<tr>
<td>_CodeFeature-Res01</td>
<td>CHARACTER</td>
<td>Reserved for future use.</td>
</tr>
<tr>
<td>_CodeFeature-Res02</td>
<td>INTEGER</td>
<td>Reserved for future use.</td>
</tr>
</tbody>
</table>

### Table 26–21: Database connection file (_Connect)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Connect-2phase</td>
<td>INTEGER</td>
<td>Displays a two-phase commit flag. “1” indicates that 2-phase commit is enabled. “0” indicates that it is not enabled.</td>
</tr>
<tr>
<td>_Connect-Batch</td>
<td>CHARACTER</td>
<td>Displays batch users status.</td>
</tr>
<tr>
<td>Field name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_Connect-CacheInfoType</td>
<td>CHARACTER</td>
<td>Displays the client database-request statement cache types, which include:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ABL Program</strong> — Current ABL program and line number.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ABL Stack</strong> — 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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>SQL</strong> — SQL statement.</td>
</tr>
<tr>
<td>_Connect-CacheInfo[32]</td>
<td>CHARACTER</td>
<td>Contains tracing information for:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ABL Program</strong> — Extent 1 contains the fully qualified ABL program with procedure name if applicable.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ABL Stack</strong> — Each extent contains a fully qualified ABL program name. Only the most current 32 program names are returned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>SQL Statement</strong> — Extent 1 contains a SQL statement.</td>
</tr>
<tr>
<td>_Connect-CacheLineNumber[32]</td>
<td>INTEGER</td>
<td>Contains line numbers. Based on the CacheInfoType setting, this field includes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ABL program</strong> — Extent 1 contains the line number of the current ABL program.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>ABL Stack</strong> — Each extent contains the line number of the fully qualified ABL program name. Only the most current 32 program names are returned.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• <strong>SQL Statement</strong> — 0.</td>
</tr>
<tr>
<td>_Connect-CacheLastUpdate</td>
<td>CHARACTER</td>
<td>The current date and time of the client database-request statement cache.</td>
</tr>
</tbody>
</table>
Table 26–21: Database connection file (_Connect)  (3 of 4)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Connect-CachingType</td>
<td>INTEGER</td>
<td>The type of client database-request statement caching performed by the client. This field includes:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1 — One ABL program or a single SQL statement will be stored in the client database-request statement cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2 — An ABL program stack or a single SQL statement stored in the client database-request statement cache.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 3 — The client sends an ABL program stack or a single SQL statement to the client database-request statement cache.</td>
</tr>
<tr>
<td>_Connect-ClientType</td>
<td>CHARACTER</td>
<td>This client type for Release 10.1B and higher remote and local clients connected to the database. The field will not have a value if the client is from before Release 10.1B. The field may contain one of the following values:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• ABL — ABL client</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SQLC — SQL client</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• WTA — Webspeed agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• APSV — AppServer agent</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• SQFC — SQL Federated client</td>
</tr>
<tr>
<td>_Connect-Device</td>
<td>CHARACTER</td>
<td>Device from which the user has connected to the database.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>_Connect-Disconnect</td>
<td>INTEGER</td>
<td>Displays a disconnect flag.</td>
</tr>
<tr>
<td>_Connect-Interrupt</td>
<td>INTEGER</td>
<td>Displays an interrupt flag.</td>
</tr>
<tr>
<td>_Connect-IPAddress</td>
<td>CHARACTER</td>
<td>Displays the IP address of a remote client connection.</td>
</tr>
<tr>
<td>_Connect-Name</td>
<td>CHARACTER</td>
<td>Displays the user name used for the connection.</td>
</tr>
<tr>
<td>_Connect-Pid</td>
<td>INTEGER</td>
<td>Displays the process ID of the user session.</td>
</tr>
</tbody>
</table>
### Table 26–21: Database connection file (._Connect) (4 of 4)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Connect-Resync</td>
<td>INTEGER</td>
<td>Displays a flag that indicates the database is resyncing.</td>
</tr>
<tr>
<td>_Connect-SemId</td>
<td>INTEGER</td>
<td>Displays the semaphore ID.</td>
</tr>
<tr>
<td>_Connect-SemNum</td>
<td>INTEGER</td>
<td>Displays the semaphore number.</td>
</tr>
<tr>
<td>_Connect-Server</td>
<td>INTEGER</td>
<td>Identifies the server if the connection is remote.</td>
</tr>
<tr>
<td>_Connect-Time</td>
<td>CHARACTER</td>
<td>Time the user connected to the database.</td>
</tr>
<tr>
<td>_Connect-transId</td>
<td>INTEGER</td>
<td>Displays the current transaction ID.</td>
</tr>
<tr>
<td>_Connect-Type</td>
<td>CHARACTER</td>
<td>Displays the connection type: SELF, REMC, BROK, SERV, or BTCH.</td>
</tr>
<tr>
<td>_Connect-Usr</td>
<td>INTEGER</td>
<td>Displays the user number.</td>
</tr>
<tr>
<td>_Connect-Wait</td>
<td>CHARACTER</td>
<td>Displays the shared memory resource the user is waiting on.</td>
</tr>
<tr>
<td>_Connect-Wait1</td>
<td>INT64</td>
<td>Displays additional wait flags.</td>
</tr>
</tbody>
</table>

### Table 26–22: Database features (._Database-Feature)

<table>
<thead>
<tr>
<th>Field Name</th>
<th>Data Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_DBFeature_Name</td>
<td>CHARACTER</td>
<td>Name of the database feature</td>
</tr>
<tr>
<td>_DBFeature_Enabled</td>
<td>CHARACTER</td>
<td>“1” indicates the feature is enabled; “0” indicates the feature is not enabled</td>
</tr>
<tr>
<td>_DBFeature_Active</td>
<td>CHARACTER</td>
<td>“1” indicates this feature is active within the database; “0” indicates this feature is not active</td>
</tr>
<tr>
<td>DBFeature-Res01</td>
<td>CHARACTER</td>
<td>Reserved for future use</td>
</tr>
<tr>
<td>DBFeature-Res02</td>
<td>INTEGER</td>
<td>Reserved for future use</td>
</tr>
</tbody>
</table>

### Table 26–23: Database status file (._DbStatus) (1 of 3)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_DbStatus-AiBlkSize</td>
<td>INTEGER</td>
<td>Number of bytes in an after-image block.</td>
</tr>
<tr>
<td>_DbStatus-BiBlkSize</td>
<td>INTEGER</td>
<td>Number of bytes in a before-image block.</td>
</tr>
<tr>
<td>Field name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------</td>
<td>------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_DbStatus-BiClSize</td>
<td>INTEGER</td>
<td>Number of kilobytes in the before-image cluster.</td>
</tr>
<tr>
<td>_DbStatus-BiOpen</td>
<td>CHARACTER</td>
<td>Most recent BI file open.</td>
</tr>
<tr>
<td>_DbStatus-BiSize</td>
<td>INT64</td>
<td>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.</td>
</tr>
<tr>
<td>_DbStatus-BiTrunc</td>
<td>CHARACTER</td>
<td>Time since the last truncate.</td>
</tr>
<tr>
<td>_DbStatus-CacheStamp</td>
<td>CHARACTER</td>
<td>Schema cache file time stamp.</td>
</tr>
<tr>
<td>_DbStatus-Changed</td>
<td>INTEGER</td>
<td>Flag indicating if database has changed since last backup.</td>
</tr>
<tr>
<td>_DbStatus-CIVersMinor</td>
<td>INTEGER</td>
<td>Client minor version number.</td>
</tr>
<tr>
<td>_DbStatus-Codepage</td>
<td>CHARACTER</td>
<td>The value in this field is not maintained. This field will be removed in a future release.</td>
</tr>
<tr>
<td>_DbStatus-Collation</td>
<td>CHARACTER</td>
<td>The value in this field is not maintained. This field will be removed in a future release.</td>
</tr>
<tr>
<td>_DbStatus-CreateDate</td>
<td>CHARACTER</td>
<td>Date and time of database creation.</td>
</tr>
<tr>
<td>_DbStatus-DbBlkSize</td>
<td>INTEGER</td>
<td>Database block size.</td>
</tr>
<tr>
<td>_DbStatus-DbVers</td>
<td>INTEGER</td>
<td>Database version number.</td>
</tr>
<tr>
<td>_DbStatus-DbVersMinor</td>
<td>INTEGER</td>
<td>Database minor version number.</td>
</tr>
<tr>
<td>_DbStatus-EmptyBlks</td>
<td>INT64</td>
<td>Number of empty blocks in the database.</td>
</tr>
<tr>
<td>_DbStatus-FbDate</td>
<td>CHARACTER</td>
<td>Most recent full backup.</td>
</tr>
<tr>
<td>_DbStatus-FreeBlks</td>
<td>INT64</td>
<td>Number of free blocks in the database.</td>
</tr>
<tr>
<td>_DbStatus-HiWater</td>
<td>INTEGER</td>
<td>Database blocks high-water mark.</td>
</tr>
<tr>
<td>_DbStatus-IbDate</td>
<td>CHARACTER</td>
<td>Most recent incremental backup.</td>
</tr>
<tr>
<td>_DbStatus-IbSeq</td>
<td>INTEGER</td>
<td>Sequence of last incremental backup.</td>
</tr>
<tr>
<td>_DbStatus-Integrity</td>
<td>CHARACTER</td>
<td>DbStatus-Integrity enabled flag (-1).</td>
</tr>
<tr>
<td>_DbStatus-IntFlags</td>
<td>INTEGER</td>
<td>Integrity flags.</td>
</tr>
<tr>
<td>_DbStatus-LastOpen</td>
<td>CHARACTER</td>
<td>Most recent database open.</td>
</tr>
<tr>
<td>_DbStatus-LastTable</td>
<td>INTEGER</td>
<td>Highest table number defined.</td>
</tr>
<tr>
<td>_DbStatus-LastTran</td>
<td>INTEGER</td>
<td>Last transaction ID.</td>
</tr>
</tbody>
</table>
Table 26–23: Database status file (_DbStatus)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_DbStatus-MostLocks</td>
<td>INTEGER</td>
<td>Lock table high-water mark.</td>
</tr>
<tr>
<td>_DbStatus-NumAreas</td>
<td>INTEGER</td>
<td>Number of areas.</td>
</tr>
<tr>
<td>_DbStatus-NumLocks</td>
<td>INTEGER</td>
<td>Lock table entries in use.</td>
</tr>
<tr>
<td>_DbStatus-NumSems</td>
<td>INTEGER</td>
<td>Number of semaphores.</td>
</tr>
<tr>
<td>_DbStatus-PrevOpen</td>
<td>CHARACTER</td>
<td>Previous database open.</td>
</tr>
<tr>
<td>_DbStatus-RMFreeBlks</td>
<td>INT64</td>
<td>Number of rm blocks with free space.</td>
</tr>
<tr>
<td>_DbStatus-SharedMemVer</td>
<td>INTEGER</td>
<td>Shared memory version number.</td>
</tr>
<tr>
<td>_DbStatus-ShmVers</td>
<td>INTEGER</td>
<td>Shared memory version number.</td>
</tr>
<tr>
<td>_DbStatus-Starttime</td>
<td>CHARACTER</td>
<td>Time when the database was started.</td>
</tr>
<tr>
<td>_DbStatus-State</td>
<td>INTEGER</td>
<td>State of the database.</td>
</tr>
<tr>
<td>_DbStatus-Tainted</td>
<td>INTEGER</td>
<td>Damaged database flags.</td>
</tr>
<tr>
<td>_DbStatus-TotalBlks</td>
<td>INT64</td>
<td>Number of blocks allocated to the database.</td>
</tr>
</tbody>
</table>

Table 26–24: Database file status file (_Filelist)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_FileList-BlkSize</td>
<td>INTEGER</td>
<td>Block size of the file</td>
</tr>
<tr>
<td>_FileList-Extend</td>
<td>INTEGER</td>
<td>Amount of most recent extend in blocks</td>
</tr>
<tr>
<td>_FileList-LogicalSz</td>
<td>INTEGER</td>
<td>Logical file size, in blocks</td>
</tr>
<tr>
<td>_FileList-Name</td>
<td>CHARACTER</td>
<td>Name of the file</td>
</tr>
<tr>
<td>_FileList-Openmode</td>
<td>CHARACTER</td>
<td>Displays the mode in which the file is opened</td>
</tr>
<tr>
<td>_FileList-Size</td>
<td>INTEGER</td>
<td>Size of the file</td>
</tr>
</tbody>
</table>

Table 26–25: Index statistics file (_IndexStat)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IndexStat-Blockdelete</td>
<td>INT64</td>
<td>Number of block deletes that have occurred to the index</td>
</tr>
<tr>
<td>_IndexStat-Delete</td>
<td>INT64</td>
<td>Number of times delete access has occurred to the index</td>
</tr>
<tr>
<td>_IndexStat-Create</td>
<td>INT64</td>
<td>Number of times create access has occurred to the index</td>
</tr>
</tbody>
</table>
### Table 26–25: Index statistics file (_IndexStat)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IndexStat-Read</td>
<td>INT64</td>
<td>Number of times read access has occurred to the index</td>
</tr>
<tr>
<td>_IndexStat-OsRead</td>
<td>INT64</td>
<td>Number of index reads from disk since the database was started</td>
</tr>
<tr>
<td>_IndexStat-Split</td>
<td>INT64</td>
<td>Number of split operations that have occurred to the index</td>
</tr>
</tbody>
</table>

### Table 26–26: Latch statistics file (_Latch)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Latch-Busy</td>
<td>INTEGER</td>
<td>Latch busy count</td>
</tr>
<tr>
<td>_Latch-Hold</td>
<td>INTEGER</td>
<td>User number of the last latch holder</td>
</tr>
<tr>
<td>_Latch-Lock</td>
<td>INTEGER</td>
<td>Latch lock count</td>
</tr>
<tr>
<td>_Latch-LockedT</td>
<td>INTEGER</td>
<td>Used for calculation of duration of lock</td>
</tr>
<tr>
<td>_Latch-LockT</td>
<td>INTEGER</td>
<td>Number, in microseconds, the latch was locked</td>
</tr>
<tr>
<td>_Latch-Name</td>
<td>CHARACTER</td>
<td>Latch name</td>
</tr>
<tr>
<td>_Latch-Qhold</td>
<td>INTEGER</td>
<td>User number of the last queue holder</td>
</tr>
<tr>
<td>_Latch-Spin</td>
<td>INTEGER</td>
<td>Latch spin count</td>
</tr>
<tr>
<td>_Latch-Type</td>
<td>CHARACTER</td>
<td>Latch type</td>
</tr>
<tr>
<td>_Latch-Wait</td>
<td>INTEGER</td>
<td>Latch wait (nap) count</td>
</tr>
<tr>
<td>_Latch-WaitT</td>
<td>INTEGER</td>
<td>Number, in microseconds, of wait or spin time</td>
</tr>
</tbody>
</table>

### Table 26–27: License management (_License)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Lic-ActiveConns</td>
<td>INTEGER</td>
<td>Number of active connections</td>
</tr>
<tr>
<td>_Lic-BatchConns</td>
<td>INTEGER</td>
<td>Number of current batch connections</td>
</tr>
<tr>
<td>_Lic-CurrConns</td>
<td>INTEGER</td>
<td>Number of current connections</td>
</tr>
<tr>
<td>_Lic-MaxActive</td>
<td>INTEGER</td>
<td>Maximum number of active connections</td>
</tr>
<tr>
<td>_Lic-MaxBatch</td>
<td>INTEGER</td>
<td>Maximum number of batch connections</td>
</tr>
<tr>
<td>_Lic-MaxCurrent</td>
<td>INTEGER</td>
<td>Maximum number of current connections</td>
</tr>
<tr>
<td>_Lic-MinActive</td>
<td>INTEGER</td>
<td>Minimum number of active users</td>
</tr>
</tbody>
</table>
Table 26–27: License management (_License)  

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Lic-MinBatch</td>
<td>INTEGER</td>
<td>Minimum number of batch connections</td>
</tr>
<tr>
<td>_Lic-MinCurrent</td>
<td>INTEGER</td>
<td>Minimum number of current connections</td>
</tr>
<tr>
<td>_Lic-ValidUsers</td>
<td>INTEGER</td>
<td>Number of valid license users</td>
</tr>
</tbody>
</table>

Table 26–28: Lock table status file (_Lock)¹  

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Lock-Chain</td>
<td>INTEGER</td>
<td>Chain number</td>
</tr>
<tr>
<td>_Lock-Flags</td>
<td>CHARACTER</td>
<td>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), or a queued lock (Q)</td>
</tr>
<tr>
<td>_Lock-Name</td>
<td>CHARACTER</td>
<td>User name</td>
</tr>
<tr>
<td>_Lock-RecId</td>
<td>INT64</td>
<td>RECID of the record and lock table entry</td>
</tr>
<tr>
<td>_Lock-Table</td>
<td>INTEGER</td>
<td>Table name</td>
</tr>
<tr>
<td>_Lock-Type</td>
<td>CHARACTER</td>
<td>Type of lock (for example, REC, RGET)</td>
</tr>
<tr>
<td>_Lock-Usr</td>
<td>INTEGER</td>
<td>User number of the user who owns the lock entry</td>
</tr>
</tbody>
</table>

¹ Use caution querying the _Lock table; frequent access can consume large quantities of system resources and negatively impact performance.

Table 26–29: Lock request file (_LockReq)  

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_LockReq-ExclFind</td>
<td>INT64</td>
<td>Number of exclusive lock finds acquired by the user</td>
</tr>
<tr>
<td>_LockReq-Name</td>
<td>CHARACTER</td>
<td>User name</td>
</tr>
<tr>
<td>_LockReq-Num</td>
<td>INTEGER</td>
<td>User number</td>
</tr>
<tr>
<td>_LockReq-RecLock</td>
<td>INT64</td>
<td>Number of record locks acquired by the user</td>
</tr>
<tr>
<td>_LockReq-RecWait</td>
<td>INT64</td>
<td>Number of times the user had to wait for a record lock</td>
</tr>
<tr>
<td>_LockReq-SchLock</td>
<td>INT64</td>
<td>Number of schema locks acquired by the user</td>
</tr>
<tr>
<td>_LockReq-SchWait</td>
<td>INT64</td>
<td>Number of times the user had to wait for a schema lock</td>
</tr>
</tbody>
</table>
**Table 26–29: Lock request file (_LockReq)**

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_LockReq-ShrFind</td>
<td>INT64</td>
<td>Number of share lock finds acquired by the user</td>
</tr>
<tr>
<td>_LockReq-TrnLock</td>
<td>INT64</td>
<td>Number of times a deleted record is protected until transaction end</td>
</tr>
<tr>
<td>_LockReq-TrnWait</td>
<td>INT64</td>
<td>Number of times a user waited for a record marked for deletion by an active transaction</td>
</tr>
</tbody>
</table>

**Table 26–30: Logging file (_Logging)**

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Logging-2PC</td>
<td>CHARACTER</td>
<td>“Yes” indicates that two-phase commit is enabled. “No” indicates that two-phase commit is disabled.</td>
</tr>
<tr>
<td>_Logging-2PCNickName</td>
<td>CHARACTER</td>
<td>Nickname used to identify the coordinator database. You specify a nickname when you enable two-phase commit.</td>
</tr>
<tr>
<td>_Logging-2PCPriority</td>
<td>INTEGER</td>
<td>Priority for the coordinator database. You specify the priority when you enable two-phase commit.</td>
</tr>
<tr>
<td>_Logging-AiBegin</td>
<td>CHARACTER</td>
<td>Date of the last AIMAGE BEGIN command.</td>
</tr>
<tr>
<td>_Logging-AiBlkSize</td>
<td>INTEGER</td>
<td>Size of the after-image block. You can change the AI block size. This reduces I/O rates on disks that have AI files.</td>
</tr>
<tr>
<td>_Logging-AiBuffs</td>
<td>INTEGER</td>
<td>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.</td>
</tr>
<tr>
<td>_Logging-AiExtents</td>
<td>INTEGER</td>
<td>Number of AI files or extents.</td>
</tr>
<tr>
<td>_Logging-AiGenNum</td>
<td>INTEGER</td>
<td>Generation number of the current AI extent.</td>
</tr>
<tr>
<td>_Logging-AiIO</td>
<td>CHARACTER</td>
<td>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.</td>
</tr>
<tr>
<td>_Logging-AiJournal</td>
<td>CHARACTER</td>
<td>“YES” indicates that after-imaging is enabled; “NO” indicates that after-imaging is disabled.</td>
</tr>
</tbody>
</table>
### Table 26–30: Logging file (_Logging) (2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Logging-AiLogSize</td>
<td>INTEGER</td>
<td>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.</td>
</tr>
<tr>
<td>_Logging-AiNew</td>
<td>CHARACTER</td>
<td>Date of the last AIMAGE NEW command.</td>
</tr>
<tr>
<td>_Logging-AiOpen</td>
<td>CHARACTER</td>
<td>Date when the AI log was last opened.</td>
</tr>
<tr>
<td>_Logging-BiBlkSize</td>
<td>INTEGER</td>
<td>BI block size.</td>
</tr>
<tr>
<td>_Logging-BiBuffs</td>
<td>INTEGER</td>
<td>Number of BI buffers.</td>
</tr>
<tr>
<td>_Logging-BiBytesFree</td>
<td>INTEGER</td>
<td>Number of free bytes remaining in the current BI cluster.</td>
</tr>
<tr>
<td>_Logging-BiClAge</td>
<td>INTEGER</td>
<td>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.</td>
</tr>
<tr>
<td>_Logging-BiClSize</td>
<td>INTEGER</td>
<td>BI cluster size.</td>
</tr>
<tr>
<td>_Logging-BiExtents</td>
<td>INTEGER</td>
<td>Number of BI extents.</td>
</tr>
<tr>
<td>_Logging-BiFullBuffs</td>
<td>INTEGER</td>
<td>Number of full BI buffers.</td>
</tr>
<tr>
<td>_Logging-BiIO</td>
<td>CHARACTER</td>
<td>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.</td>
</tr>
<tr>
<td>_Logging-BiLogSize</td>
<td>INTEGER</td>
<td>Number of kilobytes in the BI file.</td>
</tr>
<tr>
<td>_Logging-CommitDelay</td>
<td>INTEGER</td>
<td>Current value of the Delayed BI File Write (-Mf) parameter.</td>
</tr>
<tr>
<td>_Logging-CrashProt</td>
<td>INTEGER</td>
<td>“Yes” indicates that crash protection is enabled; “No” indicates that crash protection is disabled.</td>
</tr>
<tr>
<td>_Logging-LastCkp</td>
<td>CHARACTER</td>
<td>Time of the last checkpoint.</td>
</tr>
</tbody>
</table>

### Table 26–31: Master block file (_MstrBlk) (1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_MstrBlk-AiBlksize</td>
<td>INTEGER</td>
<td>Number of bytes in an AI block</td>
</tr>
<tr>
<td>_MstrBlk-BiBlksize</td>
<td>INTEGER</td>
<td>Number of bytes in a BI block</td>
</tr>
</tbody>
</table>
OpenEdge virtual system table schema descriptions

Table 26–31: Master block file (_MstrBlk) (2 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_MstrBlk-BiOpen</td>
<td>CHARACTER</td>
<td>Date and time the BI file was last opened</td>
</tr>
<tr>
<td>_MstrBlk-BiPrev</td>
<td>CHARACTER</td>
<td>Previous value of _MstrBlk-biopen</td>
</tr>
<tr>
<td>_MstrBlk-BiState</td>
<td>INTEGER</td>
<td>Current state of BI file</td>
</tr>
<tr>
<td>_MstrBlk-Cfilnum</td>
<td>INTEGER</td>
<td>Highest file number currently defined in the database</td>
</tr>
<tr>
<td>_MstrBlk-Crdate</td>
<td>CHARACTER</td>
<td>Date and time of database creation</td>
</tr>
<tr>
<td>_MstrBlk-Dbstate</td>
<td>INTEGER</td>
<td>Current state of the database</td>
</tr>
<tr>
<td>_MstrBlk-Dbvers</td>
<td>INTEGER</td>
<td>Database version number</td>
</tr>
<tr>
<td>_MstrBlk-Fbdate</td>
<td>CHARACTER</td>
<td>Date and time of the last full backup</td>
</tr>
<tr>
<td>_MstrBlk-Hiwater</td>
<td>INTEGER</td>
<td>Database blocks high-water mark</td>
</tr>
<tr>
<td>_MstrBlk-Ibdate</td>
<td>CHARACTER</td>
<td>Date and time of most recent incremental backup</td>
</tr>
<tr>
<td>_MstrBlk-Ibseq</td>
<td>INTEGER</td>
<td>Sequence of last incremental backup</td>
</tr>
<tr>
<td>_MstrBlk-Integrity</td>
<td>INTEGER</td>
<td>Database integrity enabled flag</td>
</tr>
<tr>
<td>_MstrBlk-Lasttask</td>
<td>INTEGER</td>
<td>Last transaction ID</td>
</tr>
<tr>
<td>_MstrBlk-Oppdate</td>
<td>CHARACTER</td>
<td>Date and time of previous database open</td>
</tr>
<tr>
<td>_MstrBlk-Oprdate</td>
<td>CHARACTER</td>
<td>Date and time of most recent database open</td>
</tr>
<tr>
<td>_MstrBlk-Rlclsize</td>
<td>INTEGER</td>
<td>Current BI cluster size</td>
</tr>
<tr>
<td>_MstrBlk-Rltime</td>
<td>CHARACTER</td>
<td>Time since last before-image file truncate</td>
</tr>
<tr>
<td>_MstrBlk-Tainted</td>
<td>INTEGER</td>
<td>Database damaged flags</td>
</tr>
<tr>
<td>_MstrBlk-Timestamp</td>
<td>CHARACTER</td>
<td>Schema cache file time stamp</td>
</tr>
<tr>
<td>_MstrBlk-Totblks</td>
<td>INT64</td>
<td>Number of blocks allocated to the database</td>
</tr>
</tbody>
</table>

Table 26–32: User connection (_MyConnection) (1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_MyConn-NumSeqBuffers</td>
<td>INTEGER</td>
<td>Number of private read-only buffers currently allowed</td>
</tr>
<tr>
<td>_MyConn-Pid</td>
<td>INTEGER</td>
<td>The user’s process ID</td>
</tr>
</tbody>
</table>
Table 26–32: User connection (_MyConnection)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_MyConn-UsedSeqBuffers</td>
<td>INTEGER</td>
<td>Number of private read-only buffers currently in use</td>
</tr>
<tr>
<td>_MyConn-UserId</td>
<td>INTEGER</td>
<td>The user’s user number</td>
</tr>
</tbody>
</table>

Table 26–33: Resource queue statistics file (_Resrc)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Resrc-Name</td>
<td>CHARACTER</td>
<td>Resource queue name</td>
</tr>
<tr>
<td>_Resrc-Lock</td>
<td>INT64</td>
<td>Resource queue lock count</td>
</tr>
<tr>
<td>_Resrc-Wait</td>
<td>INT64</td>
<td>Resource queue wait count</td>
</tr>
<tr>
<td>_Resrc-Time</td>
<td>INT64</td>
<td>Resource queue wait time</td>
</tr>
</tbody>
</table>

Table 26–34: Segments file (_Segments)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Segment-ByteFree</td>
<td>INTEGER</td>
<td>Number of free bytes in this shared memory segment</td>
</tr>
<tr>
<td>_Segment-BytesUsed</td>
<td>INTEGER</td>
<td>Number of bytes used in this shared memory segment</td>
</tr>
<tr>
<td>_Segments-Segld</td>
<td>INTEGER</td>
<td>Segment ID</td>
</tr>
<tr>
<td>_Segments-SegSize</td>
<td>INTEGER</td>
<td>Segment size</td>
</tr>
</tbody>
</table>

Table 26–35: Servers file (_Servers)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Server-CurrUsers</td>
<td>INTEGER</td>
<td>Number of current users connected to this server</td>
</tr>
<tr>
<td>_Server-Logins</td>
<td>INT64</td>
<td>Number of established logins</td>
</tr>
<tr>
<td>_Server-MaxUsers</td>
<td>INTEGER</td>
<td>Maximum number of logins available on the server</td>
</tr>
<tr>
<td>_Server-Num</td>
<td>INTEGER</td>
<td>Server number</td>
</tr>
<tr>
<td>_Server-Pid</td>
<td>INTEGER</td>
<td>Process ID of the server</td>
</tr>
<tr>
<td>_Server-PortNum</td>
<td>INTEGER</td>
<td>TCP/IP port number of the server</td>
</tr>
<tr>
<td>_Server-Protocol</td>
<td>CHARACTER</td>
<td>Protocol supported by the server</td>
</tr>
<tr>
<td>_Server-Type</td>
<td>CHARACTER</td>
<td>Server type</td>
</tr>
<tr>
<td>Field name</td>
<td>Data type</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>_Startup-AiBuffs</td>
<td>INTEGER</td>
<td>Number of after-image buffers</td>
</tr>
<tr>
<td>_Startup-AiName</td>
<td>CHARACTER</td>
<td>Value of the After-image Filename (-a) startup parameter</td>
</tr>
<tr>
<td>_Startup-Alternate_Buffs</td>
<td>INTEGER</td>
<td>Value of the Buffers in Alternate Buffer pool (-B2) startup parameter</td>
</tr>
<tr>
<td>_Startup-APWBuffs</td>
<td>INTEGER</td>
<td>Value of the Page Writer Scan (-pwscan) parameter</td>
</tr>
<tr>
<td>_Startup-APWMaxWrites</td>
<td>INTEGER</td>
<td>Value of the Page Writer Maximum Buffers (-pwmax) startup parameter</td>
</tr>
<tr>
<td>_Startup-APWQTime</td>
<td>INTEGER</td>
<td>Value of the Page Writer Queue Delay (-pwqdelay) startup parameter</td>
</tr>
<tr>
<td>_Startup-APWSTime</td>
<td>INTEGER</td>
<td>Value of the Page Writer Scan Delay (-pwsdelay) parameter</td>
</tr>
<tr>
<td>_Startup-BiBuffs</td>
<td>INTEGER</td>
<td>Number of before-image buffers</td>
</tr>
<tr>
<td>_Startup-BiDelay</td>
<td>INTEGER</td>
<td>Delay of before-image flush (-mf)</td>
</tr>
<tr>
<td>_Startup-BiIO</td>
<td>INTEGER</td>
<td>Before-image file I/O (-r, -R)</td>
</tr>
<tr>
<td>_Startup-BiName</td>
<td>CHARACTER</td>
<td>Before-image filename (-g)</td>
</tr>
<tr>
<td>_Startup-BiTrunc</td>
<td>INTEGER</td>
<td>Before-image truncate interval (-G)</td>
</tr>
<tr>
<td>_Startup-Buffs</td>
<td>INTEGER</td>
<td>Number of database buffers (-B)</td>
</tr>
<tr>
<td>_Startup-CrashProt</td>
<td>INTEGER</td>
<td>No crash protection (-i)</td>
</tr>
<tr>
<td>_Startup-Directio</td>
<td>INTEGER</td>
<td>Direct I/O data transfer (-directio)—field is set to 1 when -directio is specified</td>
</tr>
<tr>
<td>_Startup-LockTable</td>
<td>INTEGER</td>
<td>Maximum size of the locking table</td>
</tr>
<tr>
<td>_Startup-MaxClients</td>
<td>INTEGER</td>
<td>Value of the Maximum Clients per Server (-Ma) startup parameter</td>
</tr>
<tr>
<td>_Startup-MaxServers</td>
<td>INTEGER</td>
<td>Value of the Maximum Servers (-Mn) startup parameter</td>
</tr>
<tr>
<td>_Startup-MaxUsers</td>
<td>INTEGER</td>
<td>Value of the Maximum Users (-n) startup parameter</td>
</tr>
<tr>
<td>_Startup-MemOverflow</td>
<td>INT64</td>
<td>Value of the Shared Memory Overflow (-Mxs) startup parameter</td>
</tr>
<tr>
<td>_Startup-Spin</td>
<td>INTEGER</td>
<td>Number of spinlock tries before time-out occurs</td>
</tr>
</tbody>
</table>
### Table 26–37: Index and table statistics range (_StatBase)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_IndexBase</td>
<td>INTEGER</td>
<td>Index statistics base; statistics gathering begins with the index specified by _IndexBase</td>
</tr>
<tr>
<td>_TableBase</td>
<td>INTEGER</td>
<td>Table statistics base; statistics gathering begins with the table specified by _TableBase</td>
</tr>
</tbody>
</table>

### Table 26–38: Table statistics file (_TableStat)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_TableStat-Create</td>
<td>INT64</td>
<td>Number of times create access has occurred to the table</td>
</tr>
<tr>
<td>_TableStat-Delete</td>
<td>INT64</td>
<td>Number of times delete access has occurred to the table</td>
</tr>
<tr>
<td>_TableStat-Read</td>
<td>INT64</td>
<td>Number of times read access has occurred to the table</td>
</tr>
<tr>
<td>_TableStat-OSRead</td>
<td>INT64</td>
<td>Number of table reads from disk since the database was started</td>
</tr>
<tr>
<td>_TableStat-Update</td>
<td>INT64</td>
<td>Number of times update access has occurred to the table</td>
</tr>
</tbody>
</table>

### Table 26–39: Transaction file (_Trans)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Trans-Coord</td>
<td>CHARACTER</td>
<td>Name of the coordinator database</td>
</tr>
<tr>
<td>_Trans-CoordTx</td>
<td>INTEGER</td>
<td>Current coordinator transaction number</td>
</tr>
<tr>
<td>_Trans-Counter</td>
<td>INTEGER</td>
<td>Transaction count</td>
</tr>
<tr>
<td>_Trans-Duration</td>
<td>INTEGER</td>
<td>Number of seconds the transaction required to complete</td>
</tr>
<tr>
<td>_Trans-Flags</td>
<td>CHARACTER</td>
<td>Transaction flags</td>
</tr>
<tr>
<td>_Trans-Misc</td>
<td>INTEGER</td>
<td>Miscellaneous information</td>
</tr>
<tr>
<td>_Trans-Num</td>
<td>INTEGER</td>
<td>Transaction number</td>
</tr>
<tr>
<td>_Trans-State</td>
<td>CHARACTER</td>
<td>Transaction state</td>
</tr>
<tr>
<td>_Trans-Txtime</td>
<td>CHARACTER</td>
<td>Transaction start time</td>
</tr>
<tr>
<td>_Trans-Urnum</td>
<td>INTEGER</td>
<td>User number of the user running the distributed transaction</td>
</tr>
</tbody>
</table>
### Table 26–40: Transaction end lock statistics (\_TxeLock)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_Txe-Locks</td>
<td>INT64</td>
<td>Total number of times that transaction end lock is requested</td>
</tr>
<tr>
<td>_Txe-Lockss</td>
<td>INT64</td>
<td>Number of concurrently held locks (only valid for update and commit locks)</td>
</tr>
<tr>
<td>_Txe-Time</td>
<td>INTEGER</td>
<td>Not yet implemented. Always “?”</td>
</tr>
<tr>
<td>_Txe-Type</td>
<td>CHARACTER</td>
<td>Type of transaction end lock, such as update or commit</td>
</tr>
<tr>
<td>_Txe-Waits</td>
<td>INT64</td>
<td>Number of times a transaction end lock request has been queued; how many</td>
</tr>
<tr>
<td></td>
<td></td>
<td>times you had to wait</td>
</tr>
<tr>
<td>_Txe-Waitss</td>
<td>INT64</td>
<td>Number of concurrently queued locks (only valid for update and commit locks)</td>
</tr>
<tr>
<td>_Txe-Wait-Time</td>
<td>INTEGER</td>
<td>Not yet implemented. Always “?”</td>
</tr>
</tbody>
</table>

### Table 26–41: User index activity (\_UserIndexStat)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_UserIndexStat-Conn</td>
<td>INTEGER</td>
<td>User number</td>
</tr>
<tr>
<td>_UserIndexStat-Num</td>
<td>INTEGER</td>
<td>Index number</td>
</tr>
<tr>
<td>_UserIndexStat-blockdelete</td>
<td>INT64</td>
<td>Number of block deletes that have occurred to the index by the specified user</td>
</tr>
<tr>
<td>_UserIndexStat-create</td>
<td>INT64</td>
<td>Number of times create access has occurred to the index by the specified user</td>
</tr>
<tr>
<td>_UserIndexStat-delete</td>
<td>INT64</td>
<td>Number of times delete access has occurred to the index by the specified user</td>
</tr>
<tr>
<td>_UserIndexStat-read</td>
<td>INT64</td>
<td>Number of times read access has occurred to the index by the specified user</td>
</tr>
<tr>
<td>_UserIndexStat-split</td>
<td>INT64</td>
<td>Number of split operations that have occurred to the index by the specified user</td>
</tr>
</tbody>
</table>
### Table 26–42: Database input/output file (_UserIO)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_UserIO-AiRead</td>
<td>INT64</td>
<td>Number of AI read operations performed by the process</td>
</tr>
<tr>
<td>_UserIO-AiWrite</td>
<td>INT64</td>
<td>Number of AI write operations performed by the process</td>
</tr>
<tr>
<td>_UserIO-BiRead</td>
<td>INT64</td>
<td>Number of BI read operations performed by the process</td>
</tr>
<tr>
<td>_UserIO-BiWrite</td>
<td>INT64</td>
<td>Number of BI write operations performed by the process</td>
</tr>
<tr>
<td>_UserIO-DbAccess</td>
<td>INT64</td>
<td>Number of database access operations performed by the process</td>
</tr>
<tr>
<td>_UserIO-DbRead</td>
<td>INT64</td>
<td>Number of database read operations performed by the process</td>
</tr>
<tr>
<td>_UserIO-DbWrite</td>
<td>INT64</td>
<td>Number of database write operations performed by the process</td>
</tr>
<tr>
<td>_UserIO-Name</td>
<td>CHARACTER</td>
<td>User name of the process</td>
</tr>
<tr>
<td>_UserIO-Usr</td>
<td>INTEGER</td>
<td>User number of the process</td>
</tr>
</tbody>
</table>

### Table 26–43: Record locking table file (_UserLock)

(1 of 2)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_UserLock-Chain</td>
<td>INTEGER</td>
<td>Type of chain should always be REC, the record lock chain</td>
</tr>
<tr>
<td>_UserLock-Flags</td>
<td>CHARACTER</td>
<td>Any of four possible types of flags: L (LIMBO lock), P (PURGED lock entry), Q (QUEUED lock request), and U (UPGRADE request)</td>
</tr>
<tr>
<td>_UserLock-Misc</td>
<td>INTEGER</td>
<td>Miscellaneous information</td>
</tr>
<tr>
<td>_UserLock-Name</td>
<td>CHARACTER</td>
<td>User name of the process owning the lock</td>
</tr>
<tr>
<td>_UserLock-Recid</td>
<td>INT64</td>
<td>Record ID for the record being locked</td>
</tr>
</tbody>
</table>
Table 26–43:  Record locking table file (_UserLock)¹

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_UserLock-Table</td>
<td>INTEGER</td>
<td>Table number</td>
</tr>
<tr>
<td>_UserLock-Type</td>
<td>CHARACTER</td>
<td>One of the three lock types: EXCL (EXCLUSIVE-LOCK), SHR (SHARE-LOCK), or NOLK (NO-LOCK)</td>
</tr>
<tr>
<td>_UserLock-Usr</td>
<td>INTEGER</td>
<td>User number of the process owning the lock</td>
</tr>
</tbody>
</table>

¹. Use caution querying the _UserLock table; frequent access can consume large quantities of system resources and negatively impact performance.

Table 26–44:  User status (_UserStatus)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_UserStatus-Counter</td>
<td>INTEGER</td>
<td>Count blocks accessed, records moved, tables moved, or indexes compacted, etc</td>
</tr>
<tr>
<td>_UserStatus-ObjectId</td>
<td>INTEGER</td>
<td>The table or index worked on</td>
</tr>
<tr>
<td>_UserStatus-ObjectType</td>
<td>INTEGER</td>
<td>The type of object worked on</td>
</tr>
<tr>
<td>_UserStatus-Operation</td>
<td>CHARACTER</td>
<td>Name of the online utility for which status is being monitored</td>
</tr>
<tr>
<td>_UserStatus-State</td>
<td>INTEGER</td>
<td>The state the utility is in (see Table 26–45 for descriptions of utility states)</td>
</tr>
<tr>
<td>_UserStatus-Target</td>
<td>INTEGER</td>
<td>The end value of the counter (if known)</td>
</tr>
<tr>
<td>_UserStatus-UserId</td>
<td>INT64</td>
<td>User number</td>
</tr>
</tbody>
</table>

Table 26–45:  _UserStatus-State descriptions

<table>
<thead>
<tr>
<th>_UserStatus-State value</th>
<th>Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>PROUTIL</td>
<td>Attempting to get a table lock</td>
</tr>
<tr>
<td>12</td>
<td>PROUTIL</td>
<td>Attempting to get an administration lock</td>
</tr>
<tr>
<td>13</td>
<td>PROUTIL</td>
<td>Attempting to lock a storage object</td>
</tr>
<tr>
<td>24</td>
<td>PROUTIL TABLEMOVE</td>
<td>Copying records from one area to another</td>
</tr>
<tr>
<td>25</td>
<td>PROUTIL TABLEMOVE</td>
<td>Creating a new, alternate index</td>
</tr>
<tr>
<td>26</td>
<td>PROUTIL TABLEMOVE</td>
<td>Removing records from the source area after copying from source to target</td>
</tr>
</tbody>
</table>
Table 26–45: _UserStatus-State descriptions

<table>
<thead>
<tr>
<th>_UserStatus-State value</th>
<th>Utility</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>PROUTIL TABLEMOVE</td>
<td>Removing old indexes</td>
</tr>
<tr>
<td>31</td>
<td>PROUTIL IDXMOVE</td>
<td>Creating a new index</td>
</tr>
<tr>
<td>32</td>
<td>PROUTIL IDXMOVE</td>
<td>Removing old index</td>
</tr>
<tr>
<td>41</td>
<td>PROUTIL IDXCOMPACT</td>
<td>Scanning the index delete chain</td>
</tr>
<tr>
<td>42</td>
<td>PROUTIL IDXCOMPACT</td>
<td>Compacting a non-leaf index block</td>
</tr>
<tr>
<td>43</td>
<td>PROUTIL IDXCOMPACT</td>
<td>Compacting a leaf index block</td>
</tr>
<tr>
<td>51</td>
<td>PROUTIL IDXFIX</td>
<td>Scanning the record manager blocks in a database</td>
</tr>
<tr>
<td>52</td>
<td>PROUTIL IDXFIX</td>
<td>Scanning the index blocks</td>
</tr>
<tr>
<td>53</td>
<td>PROUTIL IDXFIX</td>
<td>Comparing index dbkeys for chosen indexes</td>
</tr>
<tr>
<td>54</td>
<td>PROUTIL IDXFIX</td>
<td>Building the target indexes from the source index</td>
</tr>
<tr>
<td>55</td>
<td>PROUTIL ENABLELARGFILES</td>
<td>Performing an operation on a binary large object (BLOB)</td>
</tr>
<tr>
<td>60</td>
<td>PROBKUP</td>
<td>Initializing area descriptors</td>
</tr>
<tr>
<td>61</td>
<td>PROBKUP</td>
<td>Backing up active before-image clusters</td>
</tr>
<tr>
<td>62</td>
<td>PROBKUP</td>
<td>Estimating size of backup files</td>
</tr>
<tr>
<td>63</td>
<td>PROBKUP</td>
<td>Flushing the buffer pool and before-image buffers</td>
</tr>
<tr>
<td>64</td>
<td>PROBKUP</td>
<td>Backing up area object blocks</td>
</tr>
<tr>
<td>65</td>
<td>PROBKUP</td>
<td>Backing up the before-image file</td>
</tr>
<tr>
<td>66</td>
<td>PROBKUP</td>
<td>Backing up the database blocks per area</td>
</tr>
<tr>
<td>70</td>
<td>PROUTIL DBANALYS</td>
<td>Examining the record manager chain</td>
</tr>
<tr>
<td>71</td>
<td>PROUTIL DBANALYS</td>
<td>Examining the index delete chain</td>
</tr>
<tr>
<td>72</td>
<td>PROUTIL DBANALYS</td>
<td>Examining the free chain</td>
</tr>
<tr>
<td>73</td>
<td>PROUTIL DBANALYS</td>
<td>Examining each database block</td>
</tr>
<tr>
<td>74</td>
<td>PROUTIL DBANALYS</td>
<td>Gathering fragmentation statistics</td>
</tr>
<tr>
<td><code>_UserStatus-State value</code></td>
<td>Utility</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>--------------------</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>80</td>
<td>PROUTIL DUMP</td>
<td>Validating user input</td>
</tr>
<tr>
<td>81</td>
<td>PROUTIL DUMP</td>
<td>Validating column index</td>
</tr>
<tr>
<td>82</td>
<td>PROUTIL DUMP</td>
<td>Validating column data type</td>
</tr>
<tr>
<td>83</td>
<td>PROUTIL DUMP</td>
<td>Dumping records from the specified table using the specified index</td>
</tr>
<tr>
<td>90</td>
<td>PROUTIL IDXMOVE</td>
<td>Validating user input</td>
</tr>
<tr>
<td>91</td>
<td>PROUTIL IDXMOVE</td>
<td>Validating target area</td>
</tr>
<tr>
<td>92</td>
<td>PROUTIL IDXMOVE</td>
<td>Attempting to get a table lock</td>
</tr>
<tr>
<td>93</td>
<td>PROUTIL IDXMOVE</td>
<td>Moving the index blocks from source to target areas</td>
</tr>
<tr>
<td>100</td>
<td>PROUTIL IDXFIX</td>
<td>Creating new indexes for schema index rebuilds</td>
</tr>
<tr>
<td>101</td>
<td>PROUTIL IDXFIX</td>
<td>Scanning all blocks with index blocks of interest</td>
</tr>
<tr>
<td>102</td>
<td>PROUTIL IDXFIX</td>
<td>Scanning all blocks with index or record manager blocks of interest</td>
</tr>
<tr>
<td>103</td>
<td>PROUTIL IDXFIX</td>
<td>Scanning all blocks to reconstruct the indexes</td>
</tr>
<tr>
<td>104</td>
<td>PROUTIL IDXFIX</td>
<td>Rebuilding indexes from the sort files</td>
</tr>
<tr>
<td>105</td>
<td>PROUTIL IDXFIX</td>
<td>Activating every rebuilt index</td>
</tr>
<tr>
<td>106</td>
<td>PROUTIL IDXFIX</td>
<td>Deleting old schema indexes</td>
</tr>
<tr>
<td>110</td>
<td>PROUTIL TABLEMOVE</td>
<td>Validating user input</td>
</tr>
<tr>
<td>111</td>
<td>PROUTIL TABLEMOVE</td>
<td>Validating target area</td>
</tr>
<tr>
<td>112</td>
<td>PROUTIL TABLEMOVE</td>
<td>Attempting to get a table lock</td>
</tr>
<tr>
<td>113</td>
<td>PROUTIL TABLEMOVE</td>
<td>Looking for inactive indexes</td>
</tr>
<tr>
<td>114</td>
<td>PROUTIL TABLEMOVE</td>
<td>Deleting template record from source area</td>
</tr>
<tr>
<td>115</td>
<td>PROUTIL TABLEMOVE</td>
<td>Moving table from source to target area</td>
</tr>
<tr>
<td>116</td>
<td>PROUTIL TABLEMOVE</td>
<td>Creating template record in target area</td>
</tr>
</tbody>
</table>
### Table 26–46: User table activity (_UserTableStat)

<table>
<thead>
<tr>
<th>Field name</th>
<th>Data type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>_UserTableStat-Conn</td>
<td>INTEGER</td>
<td>User number</td>
</tr>
<tr>
<td>_UserTableStat-Num</td>
<td>INTEGER</td>
<td>Table number</td>
</tr>
<tr>
<td>_UserTableStat-create</td>
<td>INT64</td>
<td>Number of times create access has occurred to the table by the specified user</td>
</tr>
<tr>
<td>_UserTableStat-delete</td>
<td>INT64</td>
<td>Number of times delete access has occurred to the table by the specified user</td>
</tr>
<tr>
<td>_UserTableStat-read</td>
<td>INT64</td>
<td>Number of times delete access has occurred to the table by the specified user</td>
</tr>
<tr>
<td>_UserTableStat-update</td>
<td>INT64</td>
<td>Updates the table</td>
</tr>
</tbody>
</table>
Index

A

_AcTActILog virtual system table 26–4
_AcTBILOg virtual system table 26–4
_AcTBuffer virtual system table 26–4
_AcTIndex virtual system table 26–4
_AcTIoFiles virtual system table 26–4
_AcTIOType virtual system table 26–4
_ActLock virtual system table 26–4
_ActOther virtual system table 26–4
_ActPWs virtual system table 26–5
_ActRecord virtual system table 26–5
_ActServer virtual system table 26–5
_ActSpace virtual system table 26–5
_ActSummary virtual system table 26–5
ADD qualifier
  PROSTRCT utility 15–8, 15–9, 15–10, 22–4
ADDONLINE qualifier
  PROSTRCT utility 22–5
-adminport startup parameter 19–10, 19–13
  hosts 24–3
  ports 3–4, 19–13, 24–3, 24–12
  Progress Explorer connecting 3–3, 24–12, 24–26
AdminServer Port (-adminport) startup parameter 19–10, 19–13
After-image (AI)
  areas 7–5, 24–25
  block size 14–22
  default 14–24
  increasing 14–24
  buffer pool 14–23
  files 6–21, 11–2, 21–105
    archiving 7–13
    managing 7–9
    marking as empty 7–15
    monitoring 7–9
    switching 7–11
  I/O 14–22, 14–23
  log 11–7
    log activity 26–4
    logging 26–7
    notes 14–16
After-image Buffers (-aibufs) startup parameter 14–23, 18–4, 19–3, 19–16
After-image extents
  backing up 4–2
  backups 4–6
  on-line backups 4–6
After-image File Management Archive
  Directory Create (-aiarcdircreate) startup parameter 19–14
After-image File Management Archive
Directory list (-aiarcdir) startup parameter 19–14

After-image File Management Archive
Interval (-aiarcinterval) startup parameter 19–15

After-image Stall (-aistall) startup parameter 7–12, 19–3, 19–16

After-image writers (AIWs) 14–16, 14–22, 14–23, 14–36, 18–4, 19–16, 19–54, 26–9

After-imaging 11–6, 13–6
disabling 7–30, 7–36, 21–22, 23–11, 23–13
enabling 7–7, 11–6, 14–25, 23–12, 23–27
roll forward recovery 6–8

-aaiarcdir startup parameter 19–14
-aaiarcdircreate startup parameter 19–14
AIARCHIVE EXTENT qualifier
RFUTIL utility 23–10
AIARCHIVER DISABLE qualifier
RFUTIL utility 23–5
AIARCHIVER ENABLE qualifier
RFUTIL utility 23–6
AIARCHIVER END qualifier
RFUTIL utility 23–7
AIARCHIVER SETARCDIR qualifier
RFUTIL utility 23–8
AIARCHIVER SETARCINTERVAL qualifier
RFUTIL utility 23–9
-aaiarcinterval startup parameter 19–15
-aibuf startup parameter 14–23, 18–4, 19–3, 19–16
AIMAGE AIOFF qualifier
RFUTIL utility 7–12, 7–36, 23–11
AIMAGE BEGIN qualifier
RFUTIL utility 7–7, 14–19, 23–11, 23–12, 23–27
AIMAGE END qualifier
RFUTIL utility 7–36, 14–24, 23–13 to ??
AIMAGE EXTENT EMPTY qualifier
RFUTIL utility 7–15, 11–6, 23–14
AIMAGE EXTENT FULL qualifier
RFUTIL utility 7–11, 7–15, 11–6, 23–14,

AIMAGE EXTENT LIST qualifier
RFUTIL utility 7–9, 7–15, 23–14, 23–16
AIMAGE EXTRACT qualifier
RFUTIL utility 23–17
AIMAGE NEW qualifier
RFUTIL utility 7–12, 23–18
AIMAGE QUERY qualifier
RFUTIL utility 23–19 to 23–20
AIMAGE SCAN qualifier
RFUTIL utility 23–21
AIMAGE TRUNCATE qualifier
RFUTIL utility 23–22
-aistall startup parameter 7–12, 19–3, 19–16
AIVERIFY qualifier 23–23, 23–24
AIWs. See After-image writers (AIWs)
AppServer 24–12
APWs. See Asynchronous page writers (APWs)
Area consistency check (-AreaCheck) startup parameter 19–17
-areacheck startup parameter 19–17
Areas. See Storage areas
.AreaStatus virtual system table 26–5
.AreaThreshold virtual system table 26–6
Asynchronous page writers (APWs) 14–11, 14–12, 14–13, 14–36, 18–5, 19–54, 26–10
activity 26–5
improving performance 14–11, 14–13, 14–17, 14–25, 19–35
monitoring 14–14
partial writes 14–16
Asynchronous replication 11–2
AUDITARCHIVE qualifier
PROUTIL utility 21–12
AUDITLOAD qualifier
PROUTIL utility 21–15
Authorization keys 21–3, 21–5, 21–29, 21–89
Auto Server (-m1) startup parameter 19–5, 19–45
B


-B2 startup parameter 19–19

Backups
after-imaging 4–9
after-imaging (AI) 7–13, 7–14
archiving 5–5
before-image files 4–2
developing strategy 6–7
important files 4–2
incremental 5–13, 24–13, 24–28
media 4–9
offline 5–6, 5–13
online 5–2, 5–8, 5–15, 23–16
operating system 5–11, 23–25
restoring 5–21
structure files 4–2
testing 5–4
types 4–3
  off-line 4–6
  on-line 4–6
  unscheduled 4–9
verifying 5–12, 5–18, 24–28

Base Index (-baseindex) startup parameter 19–7, 19–20

Base Table (-basetable) startup parameter 19–7, 19–21

-baseindex startup parameter 19–7, 19–20

-basetable startup parameter 19–7, 19–21

Batch startup commands
BPRO 21–64

Before-image (BI)
areas 24–25
block size
default 14–19
increasing 14–19
blocks 21–105
clusters 21–105
delaying writes 14–20
files 21–105
  truncating 6–25, 21–6, 21–17, 21–22, 21–88, 21–103
I/O 14–15, 14–16
log activity 26–4
logging 26–7
notes 14–16
threshold 14–21, 14–22

Before-image Block Size (-biblocksize)
startup parameter 21–105

Before-image Buffers (-bibufs)
startup parameter 14–17, 14–23, 18–6, 19–3, 19–16, 19–22

Before-image Cluster Age (-G) startup parameter 19–3, 19–37

Before-image Cluster Size (-bi) startup parameter 21–105

Before-image files
backing up 4–2

Before-image Threshold (-bithold) startup parameter 18–7

Before-image writers (BIWs) 14–15, 14–16, 14–17, 14–36, 18–6, 19–22, 19–54, 26–9
  using 14–17

BI File Write (-Mf) startup parameter 14–16

-bi startup parameter 21–105

-biblocksize startup parameter 21–105

-bibufs startup parameter 14–17, 14–23, 18–6, 19–3, 19–16, 19–22

BIGROW qualifier
PROUTIL utility 21–17

Binary large objects (BLOBs) 16–11, 21–44, 21–86

-bistall startup parameter 14–21, 19–3, 19–22, 19–23

-bithold startup parameter 14–21, 18–7, 19–3, 19–22, 19–23

BIWs. See Before-image writers (BIWs)

BLOBS. See Binary large objects (BLOBs)

_Block virtual system table 26–6

Block Access option
PROMON utility 20–11

Block Table chain 14–12

Blocking factor 24–14

Blocks
before-image (BI) 19–49
free chain 21–3, 21–28
master 22–16
record 21–3, 21–28
size 22–15

Blocks in Alternate Buffer Pool (-B2)
startup parameter 19–19
Blocks in Database Buffers (-B) startup parameter 14–9, 14–10, 14–27, 19–3, 19–18, 19–35, 21–18

-Bp startup parameter 14–10


Buffer hit 14–6

Buffer pool 14–6, 19–38

Buffered I/O (-r) startup parameter 19–59

Buffers
  after-image (AI) 14–23
  before-image (BI) 14–13, 14–15, 14–17, 19–37 increasing 14–17
database 14–5, 14–6 monitoring 14–8
tuning 14–9
empty 14–11
modified 14–8
private 26–7
private read-only 14–5, 14–9
status 26–6
BuffStatus virtual system table 26–6

BUILD INDEXES qualifier
  PROUTIL utility 16–20

BUILDDB qualifier
  PROSTRCT utility 22–7

Bulk Loader description files
  creating 16–25
  modifying 16–26
  sample 16–26

Bulk loading 16–25, 19–39, 21–18

BULKLOAD qualifier
  PROUTIL utility 16–27, 16–28, 21–18

BUSY qualifier
  PROUTIL utility 5–11, 21–19

C
  -cache startup parameter 19–24

Case Table (-cpcase) startup parameter 19–6, 19–27

Century Year Offset (-yy) startup parameter 19–11, 19–66, 21–18

Chains
  free 21–20
  RM 21–20

CHANALYS qualifier
  PROUTIL utility 21–20

Character large objects (CLOBs) 16–11, 21–23

Character sets 3–5, 21–3, 21–23, 21–26

Characters
  allowed for database names 2–16

_Checkpoint virtual system table 26–6

Checkpoints 14–17, 26–6
  -classpath startup parameter 19–10, 19–25

Clients 14–36, 19–57, 19–60

CLOBs. See Character large objects (CLOBs)

Cluster Mode (-cluster) startup parameter 19–26
  -cluster startup parameter 19–26

Clusters
  before-image (BI) 14–13, 14–15, 14–19, 14–21, 19–37, 21–3, 21–6, 21–17 increasing 14–17, 14–18 size 14–17

_Code-Feature virtual system table 26–6

CODEPAGE-COMPILED qualifier
  PROUTIL utility 19–27, 21–21

Collation Table (-cpcoll) startup parameter 19–6, 19–28

Commands
  PROAIW 18–4
  PROAPW 18–5
  PROBIW 18–6
  PROQUIET 14–21, 14–22, 18–7, 18–8
  PROSERVE 3–5, 14–37, 14–39, 18–7, 18–9, 18–10
  PROSHUT 3–12, 14–18, 14–20, 14–24, 18–4, 18–5, 18–6, 18–11, 18–13, 18–14
  PROWDOG 3–7, 18–15

Configuration Properties File (-properties) startup parameter 19–11, 19–58

_Connect virtual system table 26–6

Connecting
  over a wide area network 19–24

Connection security 8–5
<table>
<thead>
<tr>
<th>CONV910 qualifier</th>
<th>PROUTIL utility 1–17, 21–22</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONVCHAR qualifier</td>
<td>PROUTIL utility 21–23 to 21–25</td>
</tr>
<tr>
<td>Conversion Map (-convmap) startup parameter 19–6, 19–27</td>
<td></td>
</tr>
<tr>
<td>CONVFILE qualifier</td>
<td>PROUTIL utility 21–26 to 21–27</td>
</tr>
<tr>
<td>-convmap startup parameter 19–6, 19–27</td>
<td></td>
</tr>
<tr>
<td>Coordinator Information option</td>
<td>PROMON utility 20–95</td>
</tr>
<tr>
<td>-cpcase startup parameter 19–6, 19–27</td>
<td></td>
</tr>
<tr>
<td>-cpcoll startup parameter 19–6, 19–28</td>
<td></td>
</tr>
<tr>
<td>-cplog startup parameter 19–6, 19–29, 19–32</td>
<td></td>
</tr>
<tr>
<td>-cpprint startup parameter 19–6, 19–30, 19–32</td>
<td></td>
</tr>
<tr>
<td>-cppcodein startup parameter 19–6, 19–31</td>
<td></td>
</tr>
<tr>
<td>-cpstream startup parameter 19–6, 19–32, 19–33, 21–2, 22–3, 23–2</td>
<td></td>
</tr>
<tr>
<td>-cpterm startup parameter 19–6, 19–32, 19–33</td>
<td></td>
</tr>
<tr>
<td>Crashes</td>
<td>system 6–18</td>
</tr>
<tr>
<td></td>
<td>system. See System crashes</td>
</tr>
<tr>
<td>CRC codes 5–19</td>
<td></td>
</tr>
<tr>
<td>CRC. See Cyclic redundancy check (CRC)</td>
<td></td>
</tr>
<tr>
<td>CREATE qualifier</td>
<td>PROSTRCT utility 1–2, 1–3, 1–6, 1–7, 1–10, 11–6, 22–3, 22–8, 22–9, 22–10</td>
</tr>
<tr>
<td>Creating incremental data definitions files 16–7 structure description (.st) files 1–3</td>
<td></td>
</tr>
<tr>
<td>Cyclic redundancy check (CRC) 21–43, 21–86, 21–89</td>
<td></td>
</tr>
</tbody>
</table>

**D**

| Data collision 11–3, 11–5 |
| Data compression 24–15, 24–16 |
| Data consolidation model 11–3, 11–4 |
| Data distribution model 11–3 |
| Data extents backing up 4–2 |
| Data ownership models 11–3 |
| See also Data distribution model, Data consolidation model, Peer-to-peer model |
| Data Types limits 2–19 |
| Data Values limits 2–19 |
| Database connecting to over a wide area network 19–24 |
| Database consistency check (-DbCheck) startup parameter 19–34 |
| Database files backing up 4–2 |
| Database limits block sizes 2–2 index limits 2–10 number of sequences 2–11 number of simultaneous transactions 2–15 number of users 2–14 records per block 2–3 size 2–13 |
| Database names characters not used 2–16 |
| Database Service Communication Area Size (-pica) startup parameter 19–58 |
| Database Status option PROMON utility 20–21 |
| Database-Feature virtual system table 26–6 |
| Databases backing up See Backups backups types. See Backups backwards compatibility 21–43, 21–86 buffer activity 26–4 converting 1–2, 1–17, 21–5, 21–22 |
copying 1–26
creating 1–2, 16–30
   Data Administration tool 1–15
   Data Dictionary tool 1–15, 24–24
   PRODB 1–13
   PROSTRCT CREATE 1–3
damaged 6–27, 6–28, 6–29, 6–30, 22–16
deleting 1–28
dumping contents 6–29
   indexes 15–25
   tables 15–25
   modes
   buffered 21–84
   multi-user 21–4, 21–19, 21–63, 21–84, 23–11
   single-user 21–4, 21–19, 21–63, 23–4
   unbuffered 21–84
   nicknames 13–7
   online 22–11
   quiet points. See Quiet points
   reconstructing 23–26
   recovering 6–2, 6–6, 6–18, 6–20
   recovery plans 6–9
   restoring 5–20, 5–22, 6–20, 6–21
      full 5–22
      incremental 5–23
   roll-forward recovery 6–4, 6–17, 7–30
   dependency 12–9
   source 24–21
   starting 3–4
      See also Brokers, Servers
      statistics 21–5
      stopping 3–12, 3–16
      table limits 2–10
      target 16–2, 16–31, 24–21, 24–28
   unlocking 6–28

DataServers
   ODBC 24–12
   ORACLE 24–12

_DELAY utility 16–9, 16–10, 16–11, 21–42 to 21–44
_DELAY qualifier
   _DbStatus virtual system table 26–6
   DBTOOL utility 24–7 to 24–10
   Delay BI File Write (-Mf) startup parameter
      14–20
   Delayed BI File Write (-Mf) startup parameter 14–20, 19–4, 19–37, 19–49
   Dependency 12–9
   DESCRIBE qualifier
      PROUTIL utility 21–30 to 21–33
   Direct I/O (-directio) startup parameter
      14–25, 19–3, 19–35
   -directio startup parameter 14–25, 19–3, 19–35
   DISABLEAUDITING qualifier
      PROUTIL utility 21–35
   DISABLEB2 qualifier
      PROUTIL utility 21–36
   DISABLEENCRYPTION qualifier
      PROUTIL utility 21–37
   DISABLEJTA qualifier
      PROUTIL utility 21–39
   DISABLEKEYEVENTS qualifier
      PROUTIL Utility 21–40
   Disk mirroring 5–9
   Disk space 7–4
   Disks
      full 6–23
   DISPTOSSCREATELIMITS qualifier
      PROUTIL utility 21–41
   DUMP qualifier
      PROUTIL utility 16–9, 16–10, 16–11, 21–42 to 21–44
   Dumping
      auto-connect records 16–8
      binary 16–9, 21–4, 21–42, 21–45
      binary by field 21–4
      contents 16–9
      definitions 16–2, 16–3 to 16–4
      field contents 16–11
      sequence definitions 16–7
      sequence values 16–16
      SQL tables 25–2
      SQL view contents 16–17
      table contents 16–2, 16–13, 16–13 to 16–14
      user table content 16–16
Dumping and loading 14–41
  binary 16–2, 16–20
  aborted 16–21
  economizing disk space 16–31
  limitations 16–2
  manually 16–2
  optimizing data access 16–32 to 16–33
  overview 16–2
  special purposes 16–30
DUMPSPECIFIED qualifier
  PROUTIL utility 16–11, 21–45 to 21–46

E
  -ecsize startup parameter 19–35
ENABLEAUDITING qualifier
  PROUTIL utility 21–47
ENABLEB2 qualifier
  PROUTIL utility 21–49
ENABLEENCRYPTION qualifier
  PROUTIL utility 21–50
ENABLEJTA qualifier
  PROUTIL utility 21–52
ENABLEKEYEVENTS qualifier
  PROUTIL utility 21–53
ENABLELARGEFILES qualifier
  PROUTIL utility 21–54, 21–56
ENABLELARGEKEYS qualifier
  PROUTIL utility 21–55
ENABLESTOREDPROC qualifier
  PROUTIL utility 21–57
Encryption cache size (-ecsize) startup parameter 19–35
EPOLICY INFO qualifier
  PROUTIL utility 21–58
EPOLICY MANAGE qualifier
  PROUTIL utility 21–59
EPOLICY SCAN qualifier
  PROUTIL utility 21–61
EPOLICY VIEW qualifier
  PROUTIL utility 21–62
Error-correction blocks 5–19
Event Level (-evtlevel) startup parameter
  17–13, 19–3, 19–36
  -evtlevel startup parameter 17–13, 19–3, 19–36
  See also After-image extents
  after-image (AI) 1–6, 7–2, 7–5, 7–15,
  11–6, 11–7, 19–16, 22–10, 23–11,
  23–14, 23–17, 23–18, 23–22
  busy 7–9
  empty 7–9
  full 7–9
  application data (Dn) 1–6, 22–10
  before-image (BI) 1–6, 1–7, 14–17,
  19–23, 21–102, 22–9
  creating 1–7
  data 22–14
  fixed-length 1–7, 7–3, 22–12
  name 22–11
  number 22–11
  schema 1–6, 1–7, 22–10
  size 22–11
  tokens
    after-image (ai) 22–12
    before-image (bi) 22–12
    data (d) 22–12
    transaction log (tl) 22–12
    transaction log (TL) 1–6, 22–10, 23–30,
    24–13
    type 22–11
    variable-length 1–8, 7–3, 22–12

F
Failover clusters 12–1
  cluster dependency 12–9
  cluster resource 12–9
  nodes 12–2
  policy 12–9
  resource group 12–9
Failures
  client machine 13–15
  network communication 13–14
  power 13–15
  servers 13–14
File handles
  maximum number 2–17
File size limitations 21–43
File tokens 7–5
  _Filelist virtual system table 26–6
Files
  after-image (.ai) 11–2, 19–16, 21–83,
  21–105, 23–19, 23–21
  application data (.dn) 21–83
  backing up 4–2
truncating 14–24
binary dump (.bd) 16–10
Bulk Loader Description (.fd) 16–25, 16–26, 16–27, 21–18
code page (.cp) 21–24
conversion map 19–27, 21–3, 21–21, 21–23
data (.d) 16–14, 19–32, 21–3
data definition (.df) 1–12, 16–3, 21–18
format 16–5
incremental 16–6
database (.db) 1–12, 21–83, 22–14
database log (.lg) 16–21
descriptors 14–39
event log (.lg) 17–2
index rebuild (.xb) 14–48
large 21–4
sort (.srt) 14–48
SQL dump (.dsq) 25–3, 25–7
structure description (.st) 1–7, 22–3, 22–11, 22–14
text dump (.d) 16–13
word-rules 21–6
Firewalls 19–50
Fixed-length extents
on-line backups 4–6
4GL tools
See Data Administration tool, Data Dictionary tool
Fragmentation
analyzing 14–40
eliminating 14–41
Full backup 4–4

H
-H startup parameter 3–6, 19–10, 19–38, 19–60
-hash startup parameter 14–9, 19–3, 19–38
Hash Table Entries (-hash) startup parameter 14–9, 19–3, 19–38
Hex values 21–24
HOLDER qualifier
PROUTIL utility 21–63
Host Name (-H) startup parameter 3–6, 19–10, 19–38, 19–60

I
-i startup parameter 14–21, 16–34, 19–3, 19–39
IDXACTIVATE qualifier
PROUTIL utility 21–64
IDXANALYS qualifier
PROUTIL utility 14–44, 14–45, 15–30, 16–9, 21–65, 21–73
IDXBUILD qualifier
IDXCHECK qualifier
PROUTIL utility 21–70 to 21–72
IDXCOMPACT qualifier
IDXFIX qualifier
PROUTIL utility 15–30, 21–69, 21–75 to 21–78
IDXMOVE qualifier
increase startup parameters 19–3
increase startup parameters online 21–81
Incremental backups 4–4
Index consistency check (-IndexCheck)
startup parameter 19–40
Index Range Size (-indexrangesize) startup parameter 19–7, 19–20, 19–41

G
-G startup parameter 19–3, 19–37
Group Delay (-groupdelay) startup parameter 14–20, 19–3, 19–37
-groupdelay startup parameter 14–20, 19–3, 19–37
-IndexCheck startup parameter 19–40

Indexes 26–6, 26–7
activating 21–68
activity 26–4
B-trees 14–10, 14–45
compacting 14–45, 15–27, 21–4, 21–69, 21–73
corrupt 14–46, 21–5, 21–68, 21–70
deactivating 16–34
deleting 21–103
moving 15–26, 21–5, 21–79
reactivating 14–51
rebuilding 14–46, 14–50, 21–5
repairing 21–5, 21–69
scanning 21–75
size limitations 14–48
statistics 26–7
use 14–44
analyzing 14–44
percentage 14–44

-indexrangesize startup parameter 19–7, 19–20, 19–41

_IndexStat virtual system table 26–6


Internet Protocol (-ipver) startup parameter 19–41

IOSTATS qualifier
PROUTIL utility 21–83 to 21–84

-ipver startup parameter 19–41

K

Key Alias (-keyalias) startup parameter 19–9, 19–42

Key Alias Password (-keyaliaspasswd) startup parameter 19–9, 19–43

Key store
backing up 4–2

Key Store Passphrase (-Passphrase) startup parameter 19–56

-keyalias startup parameter 19–9, 19–42

-keyaliaspasswd startup parameter 19–9, 19–43

L

-L startup parameter 19–43

_Latch virtual system table 26–6

Latches 14–39, 19–63, 26–6

Least recently used (LRU) 14–10

Least recently used (LRU) 14–9, 14–10, 14–13

_License virtual system table 26–6

Limits

database
number of simultaneous transactions 2–15
number of users 2–14
database names 2–16
for database names 2–16

LIST qualifier
PROSTRCT utility 1–14, 1–27, 11–6, 15–6, 15–8, 15–9, 15–10, 15–14, 15–24, 22–4, 22–11
PROSTRCT utility 1–12

-lkhash startup parameter 19–44

-lkrela
startup parameter 19–3

LOAD qualifier
PROUTIL utility 21–85 to 21–86

Load records
bad 16–29
reconstructing 16–29

Loading
binary 21–5, 21–85
bulk See Bulk loading
contents 16–20 to 16–24
definitions 16–18 to 16–19
4GL 16–18
table 16–18
no-integrity mode 16–34
sequence values 16–24
SQL tables 25–6
SQL view contents 16–23
table contents
Data tool 16–22
PROUTIL 16–20
PROUTIL BULKLOAD 16–27
user table contents 16–23

_Lock virtual system table 26–6

Index–9
Lock Table Entries (-L) startup parameter 19–3, 19–43
Lock Table Hash Size (-lkhash) startup parameter 19–44
Locking and Waiting statistics option PROMON utility 20–9
_LockReq virtual system table 26–7
Locks 26–6
activity 26–4
exclusive 15–25, 19–43, 20–13, 21–102
intent exclusive 20–13
intent share 20–13
share 15–27, 19–43, 20–13, 21–79
shared on table with intent to set exclusive 20–13
transaction end 26–7
Log (.lg) files
backing up 4–2
Log File Code Page (-cplog) startup parameter 19–6, 19–29, 19–32
_Logging virtual system table 26–7
LRU. See Least recently used (LRU)

M
-m1 startup parameter 19–5, 19–45
-m2 startup parameter 19–5, 19–45
-m3 startup parameter 19–5, 19–45, 19–51
-Ma startup parameter 14–27, 19–10, 19–46, 19–50
Manual Server (-m2) startup parameter 19–5, 19–45
MARK BACKEDUP qualifier
RFUTIL utility 23–25
-maxAreas startup parameter 19–46
Maximum Minimum Servers (-Mn) startup parameter 19–50
Maximum area number (-maxAreas) startup parameter 19–46
Maximum Clients per Server (-Ma) startup parameter 14–27, 19–10, 19–46, 19–50
Maximum Dynamic Server (-maxport) startup parameter 19–10, 19–47, 19–50
Maximum JTA Transactions (-maxxids) startup parameter 19–47
Maximum Number of Users (-n) startup parameter 19–46, 19–50
Maximum Servers (-Mn) startup parameter 14–27, 14–38, 19–10, 19–46, 19–51
Maximum Servers per Broker (-Mpb) startup parameter 19–10, 19–52
-maxport startup parameter 19–10, 19–47, 19–50
MAXUP UNIX parameter 14–36
-maxxids startup parameter 19–47
-MemCheck startup parameter 19–48
Memory monitoring 14–28
Memory consistency check (-MemCheck) startup parameter 19–48
Merge Number (-TM) startup parameter 14–50, 14–51
-Mf startup parameter 14–16, 14–20, 19–4, 19–37, 19–49
-Mi startup parameter 19–10, 19–46, 19–50, 19–51
Minimum Clients per Server (-Mi) startup parameter 19–10, 19–46, 19–50, 19–51
Minimum Dynamic Server (-minport) startup parameter 19–10, 19–47, 19–50
-minport startup parameter 19–10, 19–47, 19–50
Modify Defaults option
PROMON utility 20–97
Most recently used (MRU) 14–10
-Mp startup parameter 19–10, 19–51
-Mpb startup parameter 19–10, 19–52
-Mpte startup parameter 19–4, 19–52
MRU. See Most recently used (MRU)
_MstrBlk virtual system table 26–7
-mux startup parameter 19–52
MVSCH qualifier
  PROUTIL utility 1–18, 21–88
  -Mxs startup parameter 19–4, 19–53
  _MyConnection virtual system table 26–7

N
  -N startup parameter 19–10, 19–54
  NameServer 19–59, 24–12
  Nap Time Increment (-napinc) startup parameter 19–55
  Nap Time Steps (-napstep) startup parameter 19–55
  -napinc startup parameter 19–55
  -napstep startup parameter 19–55
  Network Type (-N) startup parameter 19–10, 19–54
  Networks
    connecting over wide area 19–24
  No Crash Protection (-i) startup parameter 14–21, 16–34, 19–3, 19–39
  No Session Cache (-noSessionCache) startup parameter 19–9, 19–55
  Node 12–2
  -noSessionCache startup parameter 19–9, 19–55
  NPROC UNIX parameter 14–36
  Number of Users (-n) startup parameter 14–27, 14–36, 14–38, 14–39, 18–4, 18–5, 18–6, 19–4, 19–54

O
  Off-line backups 4–6
  -omsSize startup parameter 19–56
  On-line backups 4–6
    after-image extents 4–6

P
  -P startup parameter 8–2
  Parameter file 19–2
  Parameter File (-pf) startup parameter 18–10, 19–2, 19–57, 24–24
  -Passphrase startup parameter 19–56
  Password (-P) startup parameter 8–2
  Passwords
    changing 8–8
  Pathnames
    absolute 1–14, 1–27
    relative 1–14
  Peer-to-peer model 11–3, 11–5
  -PendConnTime startup parameter 19–10, 19–57
  Pending Connection Time
    (-PendConnTime) startup parameter 19–10, 19–57
  Performance
    bottlenecks 14–2
    monitoring 14–3
  -pf startup parameter 18–10, 19–2, 19–57, 24–24
  -pica startup parameter 19–58
  Pin Shared Memory (-pinshm) startup parameter 14–27, 19–4, 19–58
  -pinshm startup parameter 14–27, 19–4, 19–58
  Policy 12–9
  Primary recovery area
    size 2–12
  Primary recovery area. See Before-image (BI)
  Print Code Page (-cpprint) startup parameter 19–6, 19–30, 19–32
  Private Buffers (-Bp) startup parameter 14–10
  PROADSV utility 3–2, 3–3, 24–11 to 24–12
  PROAIW command 18–4
  PROAPW command 18–5
  PROBIW command 18–6
PROBKUP utility 5–2, 5–10, 7–11, 23–25, 24–13 to 24–16, 26–8

Procedure library (.pl) files backing up 4–2

PROCOPY utility 1–2, 1–12, 1–26, 7–6, 22–10, 24–21 to 24–22


PRODEL utility 1–28, 16–32, 24–25

Progress Explorer 3–3

Progress Explorer utility 3–2, 3–5, 18–9, 19–59, 19–61, 24–26

Progress Explorer. See Progress Explorer utility

PROLOG utility 17–4, 24–27

PROMON utility 7–4, 13–8, 13–9, 13–10, 13–16, 14–3, 18–5, 19–54, 20–2 to 20–98, 21–84, 26–1

aborting a limbo transaction 13–12
Activity option 14–9, 14–17, 14–28, 20–16, 21–34
Block Access option 20–11
committing a limbo transaction 13–12
Coordinator Information option 13–12, 20–95
Database Status option 20–21
Locking and Waiting statistics option 20–9
main menu 13–10, 13–21, 20–2
Modify Defaults option 20–97
R&D menu
Activity Displays 20–59 to 20–76
Adjust Monitor Options 20–91
Administrative Functions 14–4, 20–85 to 20–88
AI Log Activity option 14–23
BI Log Activity 14–16, 14–17
Other Displays 19–63, 20–77 to 20–81
Page Writers Activity 14–14
Shared Resources option 14–38
Shared Resources Status option 14–28
Shared-memory Segments Status option 14–28
Status Displays 20–26 to 20–45
Record Locking Table option 20–13
Resolve JTA Transactions option 20–96
Resolve Limbo Transaction option 20–94
resolve limbo transactions 13–17
resolving limbo transactions 13–12
Shared Resources option 20–19
Shut Down a Database option 14–18, 14–20, 14–24, 20–23
Shut Down Database option 18–14
transaction control 13–10 to 13–12

Transactions Control option 20–92
User Control option 20–4
-properties startup parameter 19–11, 19–58

PROQUIET command 5–9, 5–11, 14–21, 14–22, 18–7, 18–8

PROREST utility 5–4, 5–20, 5–22, 7–6, 24–28 to 24–30

PROSERVE command 3–5, 14–37, 14–39, 18–7, 18–8, 18–9, 18–10

PROSERVE utility 3–6

PROSHUT command 3–12, 14–18, 14–20, 14–24, 18–4, 18–6, 18–11, 18–13, 18–14
batch shutdown (-b) 18–11
batch shutdown with no active users (-bn) 18–11
emergency shutdown (-F) 18–11
unconditional batch shutdown (-by) 18–11

PROSTRCT utility 7–6, 15–2 to 15–11, 22–2 to 22–16
ADD qualifier 15–8, 15–9, 15–10, 22–4
ADDONLINE qualifier 22–5
BUILDDB qualifier 22–7
CREATE qualifier 1–2, 1–3, 1–6, 1–7, 1–10, 11–6, 22–3, 22–8, 22–9, 22–10
LIST qualifier 1–12, 1–14, 1–27, 11–6, 15–6, 15–8, 15–9, 15–10, 15–14, 15–24, 22–4, 22–11
REMOVE qualifier 15–11, 15–24, 22–12
REORDER AI qualifier 7–28 to 7–29, 22–13
REPAIR qualifier 5–10, 22–14
UNLOCK qualifier 22–16

Protocols 19–52
network 19–51

PROUTIL Utility
DISABLEKEYEVENTS qualifier 21–40

PROUTIL utility 13–6, 13–8, 13–9, 13–10, 13–13, 16–2, 16–9, 16–20, 16–34, 21–2, 26–8
2PHASE BEGIN qualifier 13–6, 21–7
2PHASE COMMIT qualifier 13–12, 20–95, 21–8
2PHASE END qualifier 13–8, 21–9
2PHASE MODIFY qualifier 21–10
2PHASE RECOVER qualifier 13–12, 13–14, 20–94, 21–11
AUDITARCHIVE qualifier 21–12
AUDITLOAD qualifier 21–15
BIGROW qualifier 21–17
BUILD INDEXES qualifier 16–20
BULKLOAD qualifier 16–27, 16–28, 21–18
BUSY qualifier 5–11, 21–19
CHANALYS qualifier 21–20
CODEPAGE-COMPILER qualifier 19–27, 21–21
CONV910 qualifier 1–17, 21–22
CONVCHAR qualifier 21–23 to 21–25
CONVFILE qualifier 21–26 to 21–27
DBANALYS qualifier 21–28, 21–100
DBAUTHKEY qualifier 21–29, 21–89
DBIPCS qualifier 21–34
DESCRIBE qualifier 21–30 to 21–33
DISABLEAUDITING qualifier 21–35
DISABLEB2 qualifier 21–36
DISABLEENCRIPTION qualifier 21–37
DISABLEJTA qualifier 21–39
DISABLETOSSLIMIT qualifier 21–41
DUMP qualifier 16–9, 16–10, 16–11, 21–42 to 21–44
DUMPSPECIFIED qualifier 16–11, 21–45 to 21–46
ENABLEAUDITING qualifier 21–47
ENABLEB2 qualifier 21–49
ENABLEENCRIPTION qualifier 21–50
ENABLEJTA qualifier 21–52
ENABLEKEYEVENTS qualifier 21–53
ENABLELARGEFILES qualifier 21–54, 21–56
ENABLELARGEKEYS qualifier 21–55
ENABLESTOREDPROC qualifier 21–57
EPOLICY INFO qualifier 21–58
EPOLICY MANAGE qualifier 21–59
EPOLICY SCAN qualifier 21–61
EPOLICY VIEW qualifier 21–62
HOLDER qualifier 21–63
IDXACTIVATE qualifier 21–64
IDXANALYS qualifier 14–44, 14–45, 15–30, 16–9, 21–65, 21–73
IDXCHECK qualifier 21–70 to 21–72
IDXFIX qualifier 15–30, 21–69, 21–75 to 21–78
INCREASETO qualifier 21–81
IOSTATS qualifier 21–83 to 21–84
LOAD qualifier 21–85 to 21–86
MVSCH qualifier 1–18, 21–88
RCODEKEY qualifier 21–89
REVERT qualifier 21–90
SETAREACREATELIMIT qualifier 21–93
SETAREATOSSLIMIT qualifier 21–94
SETBLOBCREATELIMIT qualifier 21–95
SETBLOTOSSSLIMIT qualifier 21–96
SETTABLECREATELIMIT qualifier 21–97
SETTABLETOSSLIMIT qualifier 21–98
TABANALYS qualifier 14–40, 16–31, 16–32, 21–99 to 21–100
TRUNCATE AREA qualifier 15–24, 21–103 to 21–104
TRUNCATE BI qualifier 14–11, 14–19, 15–24, 21–105 to 21–106
UPDATESCHEMA qualifier 21–107
UPDATEVST qualifier 21–108, 26–2
VIEWB2 qualifier 21–109
WBREAK-COMPILER qualifier 21–115
WORD-RULES qualifier 21–115
PROWDOG command 3–7, 18–15, 19–54

Q
Quiet points 5–9

R
-r startup parameter 19–59
R-code 16–7
R-code in Code Page (-cprcodein) startup parameter 19–6, 19–31
RCODEKEY qualifier
PROUTIL utility 21–89
Record Locking Table option
PROMON utility 20–13
Recovery
roll-forward 7–30, 13–6
Recovery plans
developing 6–7
REMOVE qualifier
PROSTRCT utility 15–11, 15–24, 22–12
REORDER AI qualifier
PROSTRCT utility 7–28 to 7–29, 22–13
REPAIR qualifier
PROSTRCT utility 5–10, 22–14
Server Group (-servergroup) startup parameter 19–11, 19–61
Server groups 19–58
Server startup parameters
  statistic collection specification 19–8
  -servergroup startup parameter 19–11, 19–61
    accessing 19–50
    activity 26–5
    performance 14–4
    remote 19–45
    stopping 3–12
Servers per Protocol (-Mp) startup parameter 19–10, 19–51
  _Servers virtual system table 26–7
Service Name (-S) startup parameter 3–6, 19–11, 19–59
Session Timeout (-sessiontimeout) startup parameter 19–9, 19–61
  -sessiontimeout startup parameter 19–9, 19–61
SETAREACREATELIMIT qualifier
  PROUTIL utility 21–93
SETAREATOSSLIMIT qualifier
  PROUTIL utility 21–94
SETBLOBCREATELIMIT qualifier
  PROUTIL utility 21–95
SETBLOBTOSSLIMIT qualifier
  PROUTIL utility 21–96
SETTABLECREATELIMIT qualifier
  PROUTIL utility 21–97
SETTABLETOSSLIMIT qualifier
  PROUTIL utility 21–98
-SG startup parameter 14–50
Shared device 12–2
Shared memory segment size (-shmsegsize)
  startup parameter 19–62
  -shmsegsize startup parameter 19–62
Shared Resources option
  PROMON utility 20–19
  Shared-memory Overflow Size (-Mxs)
    startup parameter 19–4, 19–53
  -shmsegsize startup parameter 19–62
  Shut Down Database option
    PROMON utility 20–23
  Single-user Progress
    on-line backups 4–6
  Sort (SRT)
    blocks 14–50
    files 14–48, 14–50
  Sort Grouping (-SG) startup parameter 14–50
  Space allocation 26–5
  Speed Sort (-TB) startup parameter 14–50
  Spin Lock Retries (-spin) startup parameter 14–4, 14–39, 19–4, 19–63
  Spin locks 14–4, 14–39
  -spin startup parameter 14–4, 14–39, 19–63
  Spinning 14–39
  SQL Server Java Classpath (-classpath)
    startup parameter 19–10, 19–25
  SQLDUMP utility 25–2 to 25–5
  SQLLOAD utility 25–6 to 25–9
  SQLSCHEMA utility 25–10 to 25–12
  SRT. See Sort
  SSL (-ssl)
    startup parameter 19–9, 19–64
    SSL connections 3–8 to 3–9
    -ssl startup parameter 19–9, 19–64
    .st files. See Structure description (.st) files
Starting OpenEdge
  single-user
    batch mode 21–64
  Startup parameter
    Database Service Communication Area Size (-pica) 19–58
    Lock Table Hash Size (-lkhash) 19–44
    Nap Time Increment (-napinc) 19–55
    Nap Time Steps (-napstep) 19–55
    Storage Object Cache Size (-omsize) 19–56
    User MUX Latches (-mux) 19–52
Startup parameters
- AdminServer Port (-adminport) 19–10, 19–13
- After-image Buffers (-aibufs) 14–23, 18–4, 19–3, 19–16
- After-image File Management Archive Directory Create (-aiarcdircreate) 19–14
- After-image File Management Archive Directory List (-aiarcdir) 19–14
- After-image File Management Archive Interval (-aiarcinterval) 19–15
- After-image Stall (-aiistall) 7–12, 19–3, 19–16

Area consistency check (-AreaCheck) 19–17

Auto Server (-m1) 19–5, 19–45

Base Index (-baseindex) 19–7, 19–20

Base Table (-basetable) 19–7, 19–21

Before-image Block Size (-biblocksize) 21–105

Before-image Buffers (-bibufs) 14–17, 14–23, 18–6, 19–3, 19–16, 19–22

Before-image Cluster Age (-G) 19–3, 19–37

Before-image cluster size (-bi) 21–105

Before-image File Management Archive Directory Create (-aiarcdircreate) 19–14

Before-image File Management Archive Directory List (-aiarcdir) 19–14

Before-image File Management Archive Interval (-aiarcinterval) 19–15

Before-image Stall (-aiistall) 7–12, 19–3, 19–16

Before-image threshold (-bithold) 14–21, 19–3, 19–22, 19–23

Before-image threshold in alternate buffer pool (-b2) 19–19

Before-image threshold in database buffers (-B) 14–9, 14–10, 14–27, 19–3, 19–18, 19–35, 21–18

Buffered I/O (-r) 19–59

Case Table (-cpcase) 19–6, 19–27

Century Year Offset (-yy) 19–11, 19–66, 21–18

Cluster Mode (-cluster) 19–26

Collation Table (-cpcoll) 19–6, 19–28

Configuration Properties File (-properties) 19–11, 19–58

Conversion Map (-convmap) 19–6, 19–27

Database consistency check (-DbCheck) 19–34

Delayed BI File Write (-Mf) 14–20, 19–4, 19–37, 19–49

Direct I/O (-directio) 14–25, 19–3, 19–35

Encryption cache size (-ecsize) 19–35

Event Level (-eventlevel) 17–13, 19–3, 19–36

Group Delay (-groupdelay) 14–20, 19–3, 19–37

Hash Table Entries (-hash) 14–9, 19–3, 19–38

Host Name (-H) 3–6, 19–10, 19–38, 19–60

Index consistency check (-IndexCheck) 19–40

Index Range Size (-indexrangesize) 19–7, 19–20, 19–41


Key Alias (-keyalias) 19–9

Key Alias Password (-keyaliaspasswd) 19–9

Key store passphrase (-Passphrase) 19–56

Load Table Entries (-L) 19–3, 19–43

Log File Code Page (-cplog) 19–6, 19–29, 19–32

Manual Server (-m2) 19–5, 19–45

Maximum Clients per Server (-Ma) 14–27, 19–10, 19–46, 19–50

Maximum Dynamic Server (-maxport) 19–10, 19–47, 19–50

Maximum JTA Transactions (-maxxids) 19–47

Maximum Number of Users (-n) 19–50

Maximum Servers (-Mn) 14–27, 14–38, 19–10, 19–46, 19–50, 19–51

Maximum Servers per Broker (-Mpb) 19–10, 19–52

Memory consistency check (-MemCheck) 19–48

Merge Number (-TM) 14–50, 14–51

Minimum Clients per Server (-Mi) 19–10, 19–46, 19–50, 19–51

Minimum Dynamic Server (-minport) 19–10, 19–47, 19–50

Network Type (-N) 19–10, 19–54

No Crash Protection (-i) 14–21, 16–34, 19–3, 19–39

No Session Cache (-nosessioncache) 19–9

Number of Users (-n) 14–27, 14–36, 14–38, 14–39, 18–4, 18–5, 18–6, 19–4, 19–46, 19–54

Parameter File (-pf) 18–10, 19–2, 19–57, 24–24

Password (-P) 8–2

Pending Connection Time (-PendConnTime) 19–10, 19–57

Pin Shared Memory (-pinshm) 14–27, 19–4, 19–58

Print Code Page (-cpprint) 19–6, 19–30, 19–32

Private Buffers (-Bp) 14–10

R-code in Code Page (-cprcodein) 19–6, 19–31

Recovery Log Threshold (-bithold) 14–21, 19–3, 19–22, 19–23

Secondary Login Broker (-m3) 19–5, 19–45, 19–51

Semaphore Sets (-semsets) 14–38, 19–4, 19–60

Server Group (-servergroup) 19–11, 19–61

Servers per Protocol (-Mp) 19–10, 19–51

Service Name (-S) 3–6, 19–11, 19–59

Session Timeout (-sessiontimeout) 19–9
Shared-memory Overflow Size (-Mxs) 19–4, 19–53
Sort Grouping (-SG) 14–50
Speed Sort (-TB) 14–50
Spin Lock Retries (-spin) 14–4, 14–39, 19–4, 19–63
SQL Server Java Classpath (-classpath) 19–10, 19–25
SSL (-ssl) 19–9
statistic collection specification 19–8
Table consistency check (-TableCheck) 19–64
Table Range Size (-tablerangesize) 19–7, 19–21, 19–65
Temporary Directory (-T) startup parameter 14–50
Terminal Code Page (-cpterm) 19–6, 19–32, 19–33
Threshold Stall (-bistall) 14–21, 19–3, 19–22, 19–23
User ID (-U) 8–2
VLM Page Table Entry Optimization (-Mpte) 19–4, 19–52

startup parameters
increase online 19–3
_Startup virtual system table 26–7
_StatBase virtual system table 26–7
Statistic Collection startup parameters 19–8

STATISTICS qualifier
Storage areas 1–3, 2–4, 14–5, 15–8, 15–9, 15–10, 15–11, 15–24, 22–11, 22–15
after-image (AI) 7–2, 22–10, 23–12
application data (.Dn) 1–6
application data (Dn) 21–103, 22–10
before-image (BI) 18–8
creating 1–7
database control (DB) 22–3, 22–7, 22–10
number of 2–4
primary recovery 22–10
primary recovery area
size 2–12
removing 22–12
size of 2–4
transaction log (TL) 21–7, 22–10
truncating 21–5
Storage Object Cache Size (-omsize) startup parameter 19–56

Store and forward replication. See also
Synchronous replication 11–2


creating 1–3
example 1–8

Structure files
backing up 4–2

Synchronous replication 11–2

System crashes 6–18, 6–19

T

-T startup parameter 14–50

TABANALYS qualifier
PROUTIL utility 14–40, 16–31, 16–32, 21–99 to 21–100

Table consistency check (-TableCheck) startup parameter 19–64

Table Range Size (-tablerangesize) startup parameter 19–7, 19–21, 19–65

-TableCheck startup parameter 19–64

TABLEMOVE qualifier

-tablerangesize startup parameter 19–7, 19–21, 19–65

Tables
4GL 8–4
constant 16–31
deleting 21–103
fragmentation 21–5
freezing 8–10
moving 15–25, 21–5
SQL 8–4
statistics 26–7
unfreezing 8–10
word-break 21–6, 21–115

_TableStat virtual system table 26–7

-TB startup parameter 14–50

TCP. See Transmission Control Protocol

Temporary Directory (-T) startup parameter 14–50
Terminal Code Page (-cpterms) startup parameter 19–6, 19–32, 19–33

Threshold stall enabling 14–21

Threshold Stall (-bistall) startup parameter 14–21, 19–3, 19–22, 19–23

-TM startup parameter 14–50, 14–51

Tokens 1–3
  area name 1–5
  area number 1–5
  blocks per cluster 1–5
  extent size 1–5
  extent type 1–5
  records per block 1–5
  storage area type 1–5

_Trans virtual system table 26–7

Transaction log
  area 13–6, 13–8, 22–10
  extents 15–24, 24–29
  files 6–22, 6–24

Transaction log (.tl) files
  backing up 4–2

Transactions 26–7
  committed 21–8
  distributed 13–2, 13–4, 13–9
  aborted 13–13, 20–94
  case study 13–15
  committed 13–13, 13–16, 20–94
  detecting 13–9
  resolving 13–10 to 13–15
  log files 13–5
  numbers 13–3, 13–13

Transactions Control option
  PROMON utility 20–92

Transmission Control Protocol (TCP) 3–6, 3–7, 3–14

TRUNCATE AREA qualifier
  PROUTIL utility 15–24, 21–103 to 21–104

TRUNCATE BI qualifier
  PROUTIL utility 14–11, 14–19, 15–24, 21–105 to 21–106

2PHASE BEGIN qualifier
  PROUTIL utility 13–6, 21–7

2PHASE COMMIT qualifier
  PROUTIL utility 13–12

Two-phase commit 6–5, 11–2, 21–3, 21–7, 21–10, 22–12, 26–7
  deactivating 13–8
  disabling 21–9, 21–22
  enabling 13–6

2PHASE COMMIT qualifier
  PROUTIL utility 20–95, 21–8

2PHASE END qualifier
  PROUTIL utility 13–8, 21–9

2PHASE MODIFY qualifier
  PROUTIL utility 21–10

2PHASE RECOVER qualifier
  PROUTIL utility 13–12, 13–14, 20–94, 21–11

_Trans virtual system table 26–7

U

-U startup parameter 8–2

UNIX parameters
  MAXUP 14–36
  NPROC 14–36
  SEMMNI 14–38
  SEMMNS 14–38
  SEMMNU 14–38
  SEMMSL 14–38
  SHMALL 21–34

UNLOCK qualifier
  PROSTRCT utility 22–16

UPDATESCHEMA qualifier
  PROUTIL utility 21–107

UPDATEVST qualifier
  PROUTIL utility 21–108, 26–2

User Control option
  PROMON utility 20–4

User ID (-U) startup parameter 8–2

User MUX Latches (-mux) startup parameter 19–52

User reports 8–9

_UserIndexStat virtual system table 26–7

_UserIO virtual system table 26–7

_UserLock virtual system table 26–8

Users
  adding 8–5
  deleting 8–7
  IDs 8–2
  local 19–54
OpenEdge 19–54
passwords 8–2
unauthorized 8–4
validating 8–3
_UserTableStat virtual system table 26–8

Utilities
DBTOOL 24–7 to 24–10
PROADSV 3–2, 3–3, 24–11 to 24–12
PROBKUP 5–2, 5–10, 7–11, 23–25, 24–13 to 24–16, 26–8
PROCOPY 1–2, 1–12, 1–26, 7–6, 22–10, 24–21 to 24–22
PRODEL 1–28, 16–32, 24–25
Progress Explorer 24–26
PROLOG 17–4, 24–27
PROMON 7–4, 13–8, 13–9, 13–10, 13–16, 13–21, 14–3, 18–5, 19–54, 20–2 to 20–98, 21–84, 26–1
PROQUIET 5–9, 5–11
PROREST 5–4, 5–20, 5–22, 7–6, 24–28 to 24–30
PROSERVE 3–6
PROSTRCT 7–6, 15–2 to 15–11, 22–2 to 22–16
PROUTIL 13–6, 13–8, 13–9, 13–10, 13–13, 16–2, 16–9, 16–20, 16–34, 21–2, 26–8
RFUTIL 7–7, 13–6, 23–2 to 23–30
SQLDUMP 25–2 to 25–5
SQLLOAD 25–6 to 25–9
SQLSCHEMA 25–10 to 25–12

V

Variable-length extents
on-line backups 4–6
VIEWB2 qualifier
PROUTIL utility 21–109
_ActAILog 26–4
_ActBILog 26–4
_ActBuffer 26–4
_ActIndex 26–4
_ActIOFile 26–4
_ActIOType 26–4

W

Watchdog process. See PROMON command
WBREAK-COMPIILER qualifier
PROUTIL utility 21–115
WebSpeed
Messanger 24–12
Transaction Server 24–12
Wide area networks
connecting 19–24
Windows Application Event Log 17–13, 19–36
Windows Performance tool 14–3

WORD-RULES qualifier
PROUTIL utility 21–115

Y

-y startup parameter 19–11, 19–66, 21–18