PSQL v12

Advanced Operations Guide
Procedures and References for Advanced Users
disclaimer
ACTIAN CORPORATION LICENSES THE SOFTWARE AND DOCUMENTATION PRODUCT TO YOU OR YOUR COMPANY SOLELY ON AN "AS IS" BASIS AND SOLELY IN ACCORDANCE WITH THE TERMS AND CONDITIONS OF THE ACCOMPANYING LICENSE AGREEMENT. ACTIAN CORPORATION MAKES NO OTHER WARRANTIES WHATSOEVER, EITHER EXPRESS OR IMPLIED, REGARDING THE SOFTWARE OR THE CONTENT OF THE DOCUMENTATION; ACTIAN CORPORATION HEREBY EXPRESSLY STATES AND YOU OR YOUR COMPANY ACKNOWLEDGES THAT ACTIAN CORPORATION DOES NOT MAKE ANY WARRANTIES, INCLUDING, FOR EXAMPLE, WITH RESPECT TO MERCHANTABILITY, TITLE, OR FITNESS FOR ANY PARTICULAR PURPOSE OR ARISING FROM COURSE OF DEALING OR USAGE OF TRADE, AMONG OTHERS.

trademarks
Btrieve, Client/Server in a Box, and Pervasive are registered trademarks of Actian Corporation. Built on Pervasive Software, DataExchange, MicroKernel Database Engine, MicroKernel Database Architecture, Pervasive.SQL, Pervasive PSQL, Solution Network, Ultralight, and ZDBA are trademarks of Actian Corporation.
NetWare and Novell are registered trademarks of Novell, Inc. NetWare Loadable Module, NLM, Novell DOS, Transaction Tracking System, and TTS are trademarks of Novell, Inc.
Oracle, Java, all trademarks and logos that contain Oracle, or Java, are trademarks or registered trademarks of Oracle Corporation.
All other company and product names are the trademarks or registered trademarks of their respective companies.
© Copyright 2014 Actian Corporation. All rights reserved. Reproduction, photocopying, or transmittal of this publication, or portions of this publication, is prohibited without the express prior written consent of the publisher.

This product includes software developed by Powerdog Industries. © Copyright 1994 Powerdog Industries. All rights reserved. This product includes software developed by KeyWorks Software. © Copyright 2002 KeyWorks Software. All rights reserved. This product includes software developed by DUNDAS SOFTWARE. © Copyright 1997-2000 DUNDAS SOFTWARE LTD., all rights reserved. This product includes software developed by the Apache Software Foundation (http://www.apache.org/).

This product uses the free unixODBC Driver Manager as written by Peter Harvey (pharvey@codebydesign.com), modified and extended by Nick Gorham (nick@easysoft.com), with local modifications from Actian Corporation. Actian Corporation will donate their code changes to the current maintainer of the unixODBC Driver Manager project, in accordance with the LGPL license agreement of this project. The unixODBC Driver Manager home page is located at www.unixodbc.org. For further information on this project, contact its current maintainer: Nick Gorham (nick@easysoft.com).
A copy of the GNU Lesser General Public License (LGPL) is included on the distribution media for this product. You may also view the LGPL at www.fsf.org/licensing/licenses/lgpl.html.

Advanced Operations Guide
December 2014
# Contents

## About This Manual
- Who Should Read This Manual ........................................... xi
- Conventions ........................................................................... xii

## 1 PSQL Databases ................................................................. 1
- An Exploration of Object Names, Named Databases, and DSNs
  - PSQL Database Concepts .................................................. 2
  - Named Database ............................................................... 2
  - Metadata ........................................................................... 2
  - Identifiers and Object Names ............................................. 2
  - The Default Database and the Current Database ............... 5
  - File Structure ................................................................... 6
  - Access Methods ................................................................ 9
  - Client/Server Communications ......................................... 9
  - Database Code Page .......................................................... 10
  - ODBC DSN Creation Options .......................................... 10
  - Using the idhosts File ....................................................... 10

## 2 Concepts of Database Maintenance .................................. 13
- An Introduction to Database Maintenance
  - Configurations ................................................................. 14
  - Database Security ............................................................ 16
  - Data Archival and Restoration ....................................... 17
  - Troubleshooting ............................................................... 18
  - Helpful Utilities ............................................................... 19

## 3 The PSQL Component Architecture .................................. 21
- A Discussion of Architecture Features
  - PSQL Database Management System ................................. 22
  - Common Address Space ................................................... 22
  - Row Level Locking ............................................................ 22
  - MicroKernel Engine .......................................................... 22
  - Relational Engine ............................................................. 23
  - Relational Architectural Overview .................................... 25
  - PSQL Relational Architecture: Server ............................... 25
  - Error Codes ...................................................................... 27
  - PSQL Auto-Reconnect ....................................................... 28

## 4 Configuration Reference ................................................ 29
- Configuration Methods and Settings Available in PSQL
  - Configuration Overview ................................................... 30
  - Ensuring Configuration Changes Take Effect ..................... 30
  - Connecting to Different Machines ................................... 30
  - Configuration Through PCC .............................................. 32
  - Configuration Through CLI Utility ................................... 33
  - Setting a Configuration ...................................................... 33
  - Command Syntax .............................................................. 35
Contents

5 Performance ................................................................. 79
   Analyzing and Tuning Database Performance
      Analyzing Performance ........................................... 80
      Tuning Performance .............................................. 81
         Spotting Performance Bottlenecks ....................... 81
         Before You Modify Configuration Parameters ....... 82
         Minimizing Initial Connection Time ................. 82
         Maximizing Runtime Throughput ....................... 84
         Large System Cache ......................................... 89
      Performance on Hypervisor Products ................... 90

6 Database Globalization .................................................. 91
   Globalization Features of PSQL
      Overview .......................................................... 92
      Concepts and Definitions ..................................... 93
         Character Sets ................................................ 93
         Encoding ....................................................... 93
         Declaring Encodings ........................................ 94
         Collation and Sorting ....................................... 95
      Choosing a Character Set and Encoding .................. 96
      Multilingual Database Support With Unicode UTF-8
         When to Use Unicode UTF-8 ................................ 97
         Unicode UTF-8 Support in PSQL ............................ 97
         Access Methods for Unicode UTF-8 Support ............ 97

Configuration Settings Parameters ........................................ 38
   Services Configuration Parameters ............................... 39
   Server Configuration Parameters .................................. 40
      Access ............................................................ 41
      Communication Protocols ..................................... 47
      Compatibility .................................................... 49
      Data Integrity ................................................... 50
      Debugging ........................................................ 54
      Directories ........................................................ 57
      Information ....................................................... 58
      Memory Usage .................................................... 58
      Performance Tuning .............................................. 60

Windows Client Configuration Parameters .............................. 66
      Access ............................................................ 67
      Cache Engine ..................................................... 69
      Cache Engine Debugging ....................................... 71
      Communication Protocols ...................................... 71
      Performance Tuning .............................................. 72
      Security ........................................................... 74
      Application Characteristics ................................... 74

Linux Client Configuration Parameters .................................. 76
      Case of Configuration Values .................................. 76
      Client Performance Affected by “Local” Setting ....... 76
      File Names with Embedded Spaces .......................... 76
      Configuration Reference ....................................... 76
      Access ............................................................ 77
      Communication Protocols ...................................... 78
      Application Characteristics ................................... 78

Directories ................................................................. 57
Debugging ............................................................... 54
Data Integrity ........................................................... 50
Communication Protocols .............................................. 47
Access ................................................................. 41

Performance Tuning ...................................................... 60
Windows Client Configuration Parameters ................................ 66
Access ................................................................. 67
Cache Engine ........................................................... 69
Cache Engine Debugging .............................................. 71
Communication Protocols .............................................. 71
Performance Tuning ...................................................... 72
Security ............................................................... 74
Application Characteristics ........................................... 74
7 Setting Up Referential Integrity

8 PSQL Security
Contents

Migration ................................................. 173
Fault Tolerance .......................................... 174
Disaster Recovery ........................................ 175

11 PSQL and Hypervisor Products ......................... 177
Tips for Using PSQL with Various Hypervisor Products
Hypervisor Product Installation .......................... 178
Usage Topics for PSQL .................................... 179
  Physical Machine To VM Migration ...................... 179
  Configuration .......................................... 179
  VM Resource Pools and Templates ...................... 179
  Failover Cluster Support ............................... 180
  Performance .......................................... 180
  Data Backup .......................................... 180

12 Workgroup Engine in Depth ............................ 181
Technical Details and Advanced Procedures for the Workgroup Engine
Networking ................................................. 182
  NetBIOS ................................................ 182
  MicroKernel Router Decision Algorithm ................. 182
Technical Differences Between Server and Workgroup .... 183
  Platforms ............................................. 183
  User Interface ........................................ 183
  Authentication and Btrieve Security Policies .......... 183
  Gateway Support ...................................... 183
  Asynchronous I/O .................................... 183
  Default Configurations ................................ 184
  License Model ........................................ 184
Troubleshooting Workgroup Issues ....................... 185
  Time delay on first connection ........................ 185
  Status Code 116 ...................................... 185
Redirecting Locator Files ................................. 187
  Redirecting Locator File Requirements ................ 187
  Creating Redirecting Locator Files ..................... 188
  Example .............................................. 189

13 Monitoring ............................................ 191
Monitoring Your PSQL Environment ....................... 192
  Monitoring Database State ............................. 192
    Monitor Overview .................................... 202
  Command Line Interface Monitor ........................ 202
  Monitoring Data File Fragmentation ...................... 205
    Deciding When to Defragment ......................... 205
    When Defragmenter Cannot Be Used .................... 206
    Accessing Defragmenter ................................ 206
    Defragmenter GUI .................................... 207
    Defragmenter Tasks ................................... 210
    Command Line Interface Defragmenter ................ 212
  Monitoring Performance Counters ....................... 215
    Registration During Installation ...................... 215
    Data Collector Sets .................................. 215
    Using Windows Performance Monitor ................... 220
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Index Tasks</td>
<td>285</td>
</tr>
<tr>
<td>Data</td>
<td>287</td>
</tr>
<tr>
<td>Importing and Exporting ASCII File Format</td>
<td>287</td>
</tr>
<tr>
<td>Data Tasks</td>
<td>287</td>
</tr>
<tr>
<td>Btrieve Command-Line Maintenance Utility (butl)</td>
<td>291</td>
</tr>
<tr>
<td>Return Codes</td>
<td>291</td>
</tr>
<tr>
<td>Commands</td>
<td>291</td>
</tr>
<tr>
<td>Viewing Command Usage Syntax</td>
<td>292</td>
</tr>
<tr>
<td>Command Format</td>
<td>292</td>
</tr>
<tr>
<td>Command Files</td>
<td>292</td>
</tr>
<tr>
<td>Description Files</td>
<td>293</td>
</tr>
<tr>
<td>Extended File Support</td>
<td>293</td>
</tr>
<tr>
<td>Owner Names</td>
<td>294</td>
</tr>
<tr>
<td>Redirecting Error Messages</td>
<td>294</td>
</tr>
<tr>
<td>ASCII File Format</td>
<td>294</td>
</tr>
<tr>
<td>Rules for Specifying File Names on Different Platforms</td>
<td>294</td>
</tr>
<tr>
<td>Importing and Exporting Data</td>
<td>295</td>
</tr>
<tr>
<td>Copy</td>
<td>295</td>
</tr>
<tr>
<td>Load</td>
<td>296</td>
</tr>
<tr>
<td>Recover</td>
<td>297</td>
</tr>
<tr>
<td>Save</td>
<td>298</td>
</tr>
<tr>
<td>Creating and Modifying Data Files</td>
<td>301</td>
</tr>
<tr>
<td>Clone</td>
<td>301</td>
</tr>
<tr>
<td>Clrowner</td>
<td>303</td>
</tr>
<tr>
<td>Create</td>
<td>303</td>
</tr>
<tr>
<td>Drop</td>
<td>305</td>
</tr>
<tr>
<td>Index</td>
<td>306</td>
</tr>
<tr>
<td>Setowner</td>
<td>307</td>
</tr>
<tr>
<td>Sindex</td>
<td>308</td>
</tr>
<tr>
<td>Compacting Btrieve Data Files</td>
<td>309</td>
</tr>
<tr>
<td>Viewing Data File Statistics</td>
<td>310</td>
</tr>
<tr>
<td>Stat</td>
<td>310</td>
</tr>
<tr>
<td>Displaying MicroKernel Engine Version</td>
<td>313</td>
</tr>
<tr>
<td>Ver</td>
<td>313</td>
</tr>
<tr>
<td>Unloading the MicroKernel Engine and Requester (DOS only)</td>
<td>314</td>
</tr>
<tr>
<td>Stop</td>
<td>314</td>
</tr>
<tr>
<td>Performing Continuous Operations</td>
<td>315</td>
</tr>
<tr>
<td>Performing Archival Logging</td>
<td>316</td>
</tr>
<tr>
<td>16 Converting Data Files</td>
<td>321</td>
</tr>
<tr>
<td>Maintaining PSQL File Compatibility</td>
<td></td>
</tr>
<tr>
<td>Rebuild Utility Concepts</td>
<td>322</td>
</tr>
<tr>
<td>Platforms Supported</td>
<td>322</td>
</tr>
<tr>
<td>File Formats</td>
<td>323</td>
</tr>
<tr>
<td>Temporary Files</td>
<td>323</td>
</tr>
<tr>
<td>Optimizing the Rebuild Process</td>
<td>324</td>
</tr>
<tr>
<td>Log File</td>
<td>328</td>
</tr>
<tr>
<td>Rebuild Utility GUI Reference</td>
<td>329</td>
</tr>
<tr>
<td>File Options Screen</td>
<td>329</td>
</tr>
<tr>
<td>Rebuild Options Screen</td>
<td>329</td>
</tr>
<tr>
<td>Rebuild Progress Screen</td>
<td>331</td>
</tr>
<tr>
<td>Rebuild Utility Tasks</td>
<td>332</td>
</tr>
<tr>
<td>GUI Tasks</td>
<td>332</td>
</tr>
</tbody>
</table>
Contents

CLI Tasks ................................................................. 333
Command Line Parameters ........................................... 333

A Description Files ....................................................... 339
  Using Description Files to Store Btrieve File Information
    Rules for Description Files ........................................ 340
    Description File Examples ......................................... 341
    Description File Elements ......................................... 343
About This Manual

This manual describes advanced procedures and provides technical information of use to the advanced user.

Some of the information in this manual may not apply to you. For example, the chapter on Gateway engine configuration does not apply to Server engines. Such information is clearly marked throughout the manual.

Actian Corporation would appreciate your comments and suggestions about this manual. As a user of our documentation, you are in a unique position to provide ideas that can have a direct impact on future releases of this and other manuals. If you have comments or suggestions for the product documentation, post your request at the Community Forum on the PSQL website.
Who Should Read This Manual

This manual is provided for advanced users. Advanced users are considered to have a strong understanding of the underlying operating systems on which you run your PSQL-based applications. Advanced users should be comfortable configuring their operating system, and in many cases, must have administrative permissions to configure the database engine. Advanced users may include the following:

- Network administrators of networks where one or more PSQL-based applications are installed
- Value-added resellers of PSQL-based applications
- Developers of PSQL-based applications
Conventions

Unless otherwise noted, command syntax, code, and examples use the following conventions:

**CASE**

Commands and reserved words typically appear in uppercase letters. Unless the manual states otherwise, you can enter these items using uppercase, lowercase, or both. For example, you can type MYPROG, myprog, or MYprog.

**Bold**

Words appearing in bold include the following: menu names, dialog box names, commands, options, buttons, statements, and so forth.

**Monospaced font**

Monospaced font is reserved for words you enter, such as command syntax.

[ ]

Square brackets enclose optional information, as in [log_name]. If information is not enclosed in square brackets, it is required.

| |

A vertical bar indicates a choice of information to enter, as in [filename | @filename].

< >

Angle brackets enclose multiple choices for a required item, as in /D=<5 | 6 | 7>.

**variable**

Words appearing in italics are variables that you must replace with appropriate values, as in filename.

...

An ellipsis following information indicates you can repeat the information more than one time, as in [parameter...].

::=

The symbol ::= means one item is defined in terms of another. For example, a::=b means the item a is defined in terms of b.
PSQL Databases

An Exploration of Object Names, Named Databases, and DSNs

This section is divided into the following topics under the heading PSQL Database Concepts:

- Named Database
- Metadata
- Identifiers and Object Names
- The Default Database and the Current Database
- File Structure
- Access Methods
- Client/Server Communications
- Database Code Page
- ODBC DSN Creation Options
- Using the idhosts File
PSQL Database Concepts

The following topics cover PSQL database concepts:

- Named Database
- Metadata
- Identifiers and Object Names
- The Default Database and the Current Database
- File Structure
- Access Methods
- Client/Server Communications
- Database Code Page
- ODBC DSN Creation Options
- Using the idshosts File

Named Database

A named database (also called a DBname) is a database with a logical name that allows users to identify it without knowing its location. PSQL requires that all databases be named. When you name a database, you associate that name with a particular dictionary directory path and one or more data file paths.

A named database is connected to through various access methods. For ODBC access, for example, you must set up a data source name (DSN) to refer to the named database. Multiple DSNs may point to the same named database. See ODBC Database Access in ODBC Guide. For other access methods, application developers can connect to a named database using the API for that access method. Refer to the developer reference guides in the PSQL documentation.

Note: To work with named databases, you must log into the computer where the database engine is located, using an operating system user name that has administrator-level privileges or is a member of the Pervasive_Admin security group.

The easiest way to create a named database is by using PSQL Control Center. See To create a new database in PSQL User's Guide. Application developers can also create a named database through different access methods APIs. For example, see CREATE DATABASE for SQL, PvCreateDatabase() for DTI, and Data Access Application Blocks for ADO.NET.

Metadata

The Relational Engine supports two versions of metadata, referred to as version 1 (V1) and version 2 (V2). V2 metadata allows for identifier names up to 128 bytes long for many identifiers, permissions on views and stored procedures, and data dictionary files (DDFs) specific for V2 metadata.

See SQL Grammar Support in ODBC Guide.

Identifiers and Object Names

An identifier is the name of a database or of a column, table, procedure, or other named object within the database. Identifiers are designated as either regular or delimited.
Regular Identifiers

A regular identifier is an identifier that is not surrounded by double quotes. Regular identifier must begin with a letter, either upper or lower case. The remainder of the identifier can consist of any combination of upper or lower case letters, digits, and valid characters.

You cannot use a reserved word as a regular identifier.

Regular identifiers are case-insensitive.

Delimited Identifiers

A delimited identifier is an identifier surrounded by double quotes. Delimited identifier can consist of any string of valid characters enclosed in double quotes.

While it is not recommended, reserved words can be used as delimited identifiers. For example, INSERT is not permitted as a regular identifier, but "INSERT" is permitted as a delimited identifier. If an identifier is also a keyword, it must be delimited by double quotes. (For example, SELECT "password" FROM my_pword_tbl. "Password" is a keyword in the SET PASSWORD statement so it must be delimited.)

Identifier Restrictions

In addition to the general restrictions listed above, the following table lists restrictions specific to each type of identifier.

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Length Limit (bytes)</th>
<th>Invalid Characters</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Must begin with a letter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cannot be null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Must begin with a letter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Valid characters are letters, digits, and the underscore (&quot;_.&quot;)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Must begin with a letter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Name must be enclosed in double quotes</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cannot be MASTER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cannot start with UK_ if you create the index with PSQL Control Center (PCC)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If you create an index outside of PCC that starts with UK_, you cannot edit the index with PCC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Must begin with a letter</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>A foreign key and an index cannot be named the same within the same table</td>
</tr>
</tbody>
</table>
Table 1  Identifier Restrictions by Identifier Type continued

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Length Limit (bytes)</th>
<th>Invalid Characters</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>V1</strong></td>
<td><strong>V2</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Password</td>
<td>8 128</td>
<td>; ? *</td>
<td>Cannot start with a blank (space character)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Cannot be null</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Any displayable character is permissible except for those listed in the &quot;Invalid Characters&quot; column</td>
</tr>
<tr>
<td>Procedure</td>
<td>30 128</td>
<td>For regular identifiers: `~ ! @ # $ % ^ &amp; * ( ) - + = }{[[:&lt;',&gt;.,*/ ] For delimited identifiers: none</td>
<td>Name must be enclosed in double quotes</td>
</tr>
<tr>
<td>(stored)</td>
<td></td>
<td>Valid characters are letters, digits, and the underscore (&quot;_&quot;))</td>
<td>Must begin with a letter</td>
</tr>
<tr>
<td>Table</td>
<td>20 128</td>
<td>/:*?&quot;&lt;&gt;</td>
<td>(and space character)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>###</td>
<td></td>
</tr>
<tr>
<td>Trigger</td>
<td>30 128</td>
<td>For regular identifiers: `~ ! @ # $ % ^ &amp; * ( ) - + = }{[[:&lt;',&gt;.,*/ ] For delimited identifiers: none</td>
<td>Name must be enclosed in double quotes</td>
</tr>
<tr>
<td>User</td>
<td>30 128</td>
<td>/:*?&quot;&lt;&gt;</td>
<td>(and space character)</td>
</tr>
<tr>
<td>View</td>
<td>20 128</td>
<td>For regular identifiers: `~ ! @ # $ % ^ &amp; * ( ) - + = }{[[:&lt;',&gt;.,*/ ] For delimited identifiers: none</td>
<td>Name must be enclosed in double quotes</td>
</tr>
</tbody>
</table>

1Unless otherwise noted, invalid characters apply both to regular and to delimited identifiers.

2Applies to version 1 (V1) metadata. See SQL Grammar Support in ODBC Guide.

3Applies to version 2 (V2) metadata. See SQL Grammar Support in ODBC Guide.

4The names of temporary tables begin with # or ##. Therefore, # and ## are invalid characters with which to begin the name of permanent tables. See CREATE (temporary) TABLE in SQL Engine Reference.
Unique Scope

Identifiers generally must be unique within a certain scope. That is, instances of the same type of object using the same name cannot be used within the same arena. Table 2 shows the arena, or the scope, within which a given object name must be unique.

Table 2  Unique Scope for Common Identifiers

<table>
<thead>
<tr>
<th>A name for this type of object...</th>
<th>Database</th>
<th>Table</th>
<th>Stored Procedure</th>
<th>Other</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Table</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Trigger, stored procedure, user-defined functions</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User or group</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>View</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraint</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Column</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Index</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key (foreign)</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cursor</td>
<td></td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

All databases hosted by a given database engine

Cannot have the same name as a foreign key

Cannot have the same name as an index

The Default Database and the Current Database

To support existing applications that do not specify a database name when creating or opening Btrieve files, PSQL maintains the concept of a default database for each transactional database engine. The default database is a pre-defined database named “DefaultDB.” To make use of the new security models without having to modify your application code, you can associate your Btrieve data directories with the default database, then set up users and privileges in the default database to control access to the data files in those directories.

The database engine also understands the concept of the current database for each client connection. If no database name is specified in a Btrieve Login (78), Create (14), or Open (0) operation, the transactional engine assumes the operation is associated with the current database. For each client, the current database is the database to which the most recent Login (78) operation occurred (explicit login). If the client computer has requested no explicit login operations, the current database is the database to which the most recent Create (14) or Open (0) operation occurred (implicit login). If no explicit or implicit logins have occurred, then the current database is the default database, described in the preceding paragraph. Note that the current database may change at any time when the given client performs an implicit or explicit login, or closes the last file handle, making “DefaultDB” the current database. The current database for each client is independent of other clients’ activities.
PSQL Databases

The simplest way to configure the new security model for existing applications is to associate all Btrieve data directories with the default database, and set up rights for the group `PUBLIC` within this database. The group `PUBLIC` is automatically created along with the `MASTER` user when you enable security for a database. See MicroKernel Engine Security Quick Start.

**File Structure**

All PSQL databases use a common data format. This commonality allows different access methods, such as transactional and relational, to access the same data. The system through which all access methods operate is called the MicroKernel Engine.

Each PSQL database table is a separate file with a default file extension of `.MKD`. Developers, however, can specify any file name extension desired. A MicroKernel file may contain both data and indexes, and is organized into various types of pages. A MicroKernel file contains data in the common data format.

Each PSQL database also contains a set of data dictionary files, with a file extension of `.DDF`. The DDF files contain the schema of the database. The DDFs for V1 metadata and V2 metadata use different file names. See System Tables in SQL Engine Reference.

(The MicroKernel Engine is completely unconcerned with the schema of the data apart from the key fields. However, the provision for referential integrity or access via SQL requires knowledge of the schema.)

The names and locations of PSQL databases are contained in a binary file named `dbnames.cfg`. For default locations of PSQL files, see Where are the PSQL files installed? in Getting Started With PSQL.

All of the files associated with a PSQL database can be viewed from the operating system. Table 3 summarizes the associated files.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Database Names Configuration</td>
<td>The <code>dbnames.cfg</code> file. A binary file that contains the names and locations of the PSQL databases.</td>
</tr>
<tr>
<td>Data (common data format)</td>
<td>Files named, by default, <code>tablename.MKD</code> for relational databases. Each database table has a corresponding MicroKernel file. For transactional data files, the name of each file is specified by the application.</td>
</tr>
<tr>
<td>Data Dictionary</td>
<td>Files with an extension of DDF. See System Tables in SQL Engine Reference.</td>
</tr>
</tbody>
</table>

**File Size**

The size limit depends on the file version, the page size, and the number of records per page, as the following tables summarize.

**File Versions 9.5 or Newer**

The maximum size of a data file is 256 GB. You must use a file format of 9.5 or newer to have a single file size larger than 128 GB.
Note that the following table assumes no record compression on the file. If you use record compression, take into account that additional records are stored per page. See Choosing a Page Size and Estimating File Size, both in PSQL Programmer’s Guide.

Table 4  Comparisons of File Size and Page Sizes for File Versions 9.5 or Newer

<table>
<thead>
<tr>
<th>Records per Page</th>
<th>Maximum Pages (in millions)</th>
<th>File Size (GB) for Various Page Sizes (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1,024</td>
</tr>
<tr>
<td>1 - 15</td>
<td>256</td>
<td>256 GB</td>
</tr>
<tr>
<td>16 - 31</td>
<td>128</td>
<td>128 GB</td>
</tr>
<tr>
<td>32 - 63</td>
<td>64</td>
<td>64 GB</td>
</tr>
<tr>
<td>64 - 127</td>
<td>32</td>
<td>32 GB</td>
</tr>
<tr>
<td>128 - 255</td>
<td>16</td>
<td>16 GB</td>
</tr>
<tr>
<td>256 - 511</td>
<td>8</td>
<td>n/a</td>
</tr>
<tr>
<td>512 - 1023</td>
<td>4</td>
<td>n/a</td>
</tr>
<tr>
<td>1024 - 2047</td>
<td>2</td>
<td>n/a</td>
</tr>
<tr>
<td>2048 - 4095</td>
<td>1</td>
<td>n/a</td>
</tr>
</tbody>
</table>

^1"n/a" stands for "not applicable"

File Versions 9.0 or Older

The maximum size of a data file is 128 GB. You must use a file format of 9.0 or newer to have a single file size larger than 64 GB.

Note that the following table assumes no record compression on the file. If you use record compression, take into account that additional records are stored per page. See Choosing a Page Size and Estimating File Size, both in PSQL Programmer’s Guide.

Table 5  Comparisons of File Size and Page Sizes for File Versions 9.0 or Older

<table>
<thead>
<tr>
<th>File Version</th>
<th>Records per Page</th>
<th>Maximum Pages (in millions)</th>
<th>File Size (GB) for Various Page Sizes (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>512</td>
</tr>
<tr>
<td>9.0</td>
<td>1 - 15</td>
<td>256</td>
<td>128</td>
</tr>
<tr>
<td>9.0</td>
<td>16 - 31</td>
<td>128</td>
<td>64</td>
</tr>
<tr>
<td>9.0</td>
<td>32 - 63</td>
<td>64</td>
<td>32</td>
</tr>
<tr>
<td>9.0</td>
<td>64 - 127</td>
<td>32</td>
<td>16</td>
</tr>
<tr>
<td>9.0</td>
<td>128 - 255</td>
<td>16</td>
<td>n/a</td>
</tr>
<tr>
<td>8</td>
<td>any</td>
<td>16</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>any</td>
<td>16</td>
<td>16</td>
</tr>
</tbody>
</table>
**File Segmentation**

By default, a data file is automatically broken into 2 GB operating system file segments as its size passes that boundary. The configuration property, “Limit Segment Size to 2 GB,” allows you to specify whether you want files divided into 2 GB segments or unified in a single, non-segmented file. The advantage of using a larger non-segmented file is more efficient disk I/O. Therefore, you can expect increased performance.

The configuration option is part of the Performance Tuning properties for a database engine. See To access configuration settings in PCC for an engine, and Limit Segment Size to 2 GB.

The property is set to “on” by default, causing files to segment at 2 GB boundaries as with previous releases. If you set the property to “off,” files can increase past the 2 GB boundary. See also Automatic Upgrade of File Version for additional information relating to the configuration property.

Any non-segmented files are subject to the limit on file size specified by your operating system. For example, creating a large file on a FAT32 file system with Limit Segment Size to 2 GB turned “off” creates multiple 4 GB file extensions. If a previously created file is already segmented, that segmentation remains on the file.

**Automatic Upgrade of File Version**

If the configuration property “Create File Version” is set to 9.0 or higher, version 8.x files are automatically converted to version 9.0 files when they reach the file limits for version 8.x, which is 64 GB. The following table summarizes this behavior.

<table>
<thead>
<tr>
<th>Configuration Property Setting for “Create File Version”</th>
<th>Configuration Property Setting for “Limit Segment Size to 2 GB”</th>
<th>File Size At Which Automatic Upgrade of File Version Occurs</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 (default)</td>
<td>Yes (default; option check marked)</td>
<td>64 GB (the maximum size of a version 8.x file)</td>
</tr>
<tr>
<td>9 (default)</td>
<td>No (option not check marked)</td>
<td>2 GB (size at which a version 8.x file segments)</td>
</tr>
</tbody>
</table>

For example, a version 8.x file that is 5 GB in size has already passed the 2 GB segmentation boundary. Because the file is already segmented, the segmentation remains on the file. Such a file would continue to segment and grow in size until it reaches 64 GB, at which size the automatic upgrade would occur. This is true whether the configuration property is set to “yes” or “no” because the file is already segmented. As the file grows beyond 64 GB, it will continue to segment until it reaches the maximum size allowed for a version 9.0 file, 128 GB.
A version 8.x file that is 1.5 GB in size would continue to grow until it reaches 2 GB in size. At that point, the automatic upgrade occurs if the configuration property is set to “no.” The file can continue to grow as a non-segmented file up to the size limit for version 9.0 files, 128 GB. If the configuration setting is set to “yes,” the 2 GB file would continue to segment and grow until it reaches 64 GB in size. At that size, the maximum for a version 8.x file, the automatic upgrade to version 9.0 occurs. As the file grows beyond 64 GB, it will continue to segment until it reaches the maximum size allowed for a version 9.0 file, 128 GB.

The “Create File Version” option is part of the Compatibility properties for a database engine. See To access configuration settings in PCC for an engine.

Note: Automatic upgrade of file version works only for an 8.x file format to a 9.0 file format. The automatic upgrade does not work for any other combination of file versions. For example, the upgrade does not occur for an 8.x file format to a 9.5 file format, or for a 7.x file format to a 9.0 file format.

Access Methods

The two primary methods in which data is accessed from PSQL databases are transactional and relational.

With transactional, an application program navigates up and down along either physical or logical pathways through data records. Using a transactional API, an application program provides direct control and allows a developer to optimize data access based on knowledge of the underlying structure of the data. Btrieve is an example of a transactional database engine.

Relational is an access method in which data is represented as collections of tables, rows, and columns. The relational model insulates the developer from the underlying data structure and presents the data in a simple table format. ODBC is an example of a relational access method.

A single application program may include both types of access. For example, an application may use transactional access for adding and changing data, and relational access for querying the data and report writing.

You need to know the access method(s) used by the application programs that rely on your installation of PSQL. The access methods may have different configurations. You may need to customize the configuration to optimize a particular access method.

Also, troubleshooting is easier when you are aware of the access method(s) used by a given application program. For example, if an application program uses relational access through ODBC, you may need to troubleshoot a problem at the ODBC level rather than at the database management system.

See Configuration Reference for the tasks and references pertaining to customizing configurations.

Client/Server Communications

The MicroKernel Engine supports two types of processing modes, local and client/server. An application accessing the database in local mode accesses a local copy of the engine. The local engine calls upon the operating system of the workstation which performs the I/O on a local or networked hard disk.

Client/server mode uses a server MicroKernel Engine executing on a shared file server. When an application program accesses the database engine in client/server mode, the requester connects to the remote engine. This requester passes transactional-level requests and data records between the
application program and the server engine using the network protocol supported by the operating system. File I/O functions are completely handled by the server engine in client/server mode and the workstation has no operating system handles allocated to shared data files. Database manipulation is performed by the server-based engine on behalf of each workstation.

Note that the processing mode is determined by the configuration of the workstation and not the application program itself. This means that an application is capable of accessing both local and client/server database engines. The application program does not have to be recompiled to switch the application to client/server mode from local mode.

Both Workgroup and Server engine can operate in either mode. When an application on the same computer as the database engine accesses the engine, it is operating in local mode. When an application on a different machine access the engine, it is operating in client/server mode.

The client/server configurations may be customized for the Workgroup and Server versions of PSQL. Configuration settings exists in the PSQL Control Center (PCC) to facilitate the configuration of client/server configurations as well as stand-alone configurations.

See Configuration Reference for the tasks and references pertaining to configuring the client/server communications and database engine.

Database Code Page

Encoding is a standard for representing character sets. Character data must be put in a standard format, that is, encoded, so that a computer can process it digitally. An encoding must be established between the PSQL database engine (server) and a PSQL client application. A compatible encoding allows the server and client to interpret data correctly.

The encoding support is divided into database code page and client encoding. The two types of encoding are separate but interrelated. For ease of discussion, database code page and client encoding are discussed together. See Configuring Network Communications for Clients in Getting Started With PSQL.

ODBC DSN Creation Options

Refer to DSN Setup and Connection Strings in ODBC Guide. That section also discusses ODBC connection strings.

Using the idshosts File

Typically, an application provides its own file location information. As an alternative, you may provide file location mapping based on information in a text file, idshosts.

The idshosts file was one aspect of PSQL (IDS). IDS has been removed from the core product but the idshosts file is still configurable.

If your applications do not use the mapping feature through idshosts, set the configuration setting Use IDS to “Off.” If your applications already use idshosts, or if you prefer to use this alternative method to map file locations, set Use IDS to “On.” See Use IDS.

Note that performance is slower when the idshosts file is used because of the time required to access the file and read its contents.

An idshosts file may be used only with a Windows or a Linux client requester. The client may communicate with a PSQL server on Windows or Linux.
Note: PSQL 8.5 or later is required if you set Use IDS to “On.” The requester uses database URIs to represent the IDS information. Database URIs were added with PSQL 8.5. See Database URIs in PSQL Programmer’s Guide in the Developer Reference.

If Use IDS is set to “On,” you must also set Use Remote MicroKernel Engine to “On.” Use Remote MicroKernel Engine is on by default.

See Use IDS and Use Remote MicroKernel Engine.

Format of idshosts Entries
Refer to the comments in the idshosts file itself for how to format entries in the file. The comments also provide example mappings. By default, for Windows platforms, the idshosts file is installed to the \bin directory under the database client installation directory. For Linux, idshosts is installed to the \etc directory under the database client installation directory (for example, /user/local/psql/etc).
PSQL Databases
Concepts of Database Maintenance

An Introduction to Database Maintenance

PSQL v12 is a comprehensive database management system built around the MicroKernel Database Engine. PSQL offers easy installation, uncomplicated maintenance, and high levels of performance and reliability. While PSQL can run for months or years with practically no maintenance, you can get the most out of it by understanding some of its unique features and learning how to perform useful tasks. This manual describes how to tune, configure, and manage your PSQL engine and associated databases.

- Configurations
- Database Security
- Data Archival and Restoration
- Troubleshooting
- Helpful Utilities
Concepts of Database Maintenance

Configurations

You can configure separate settings for the server and client aspects of the database engine. The settings allow you to optimize the performance of the engine based on your business requirements.

The following figure illustrates the flow from an application program to the database files at the operating system. The database engine resides between the application program and the data files.

Figure 1  Conceptual View of Database Engine Configuration

The types of settings that you can configure for server include the following:

- Access
- Communication protocols
- Compatibility with previous versions of the MicroKernel Engine (MKDE)
- Data integrity
- Debugging
- Directories
- Memory usage
- Performance
The types of settings that you can configure for client include the following:

- Access
- Communication protocols
- Performance
- Security
- Application characteristics

You configure these settings within the PCC. See Configuration Reference for the tasks and references pertaining to configuration.
Database Security

The access to a PSQL database can be protected in several ways. Administrative-level security is set through the operating system. You can control who can administer a PSQL database with the security mechanisms native to the operating system.

PSQL also provides relational security at the user and group levels. You can control who can access the data and at what capability. For example, for each table within a PSQL database, you can specify whether a user or group can create, select, update, insert into, delete, or alter the table.

You establish security by setting a password for the entire database. At that point, the only user authorized to access the database is a default user named Master. You can then add additional users and groups.

Security can be set within the PCC. Also supported are two Structured Query Language (SQL) statements pertaining to security: GRANT and REVOKE. GRANT and REVOKE also allow you to set security at the column level if you choose.

The GRANT syntax integrates with transactional Owner Names, allowing owner names to be enforced when using relational access.

See Chapter 8 for information pertaining to security, Owner Names, users, and groups.
Data Archival and Restoration

Back up data is a routine part of protecting your databases and ensuring disaster recovery. You have several ways in which you can back up and restore your PSQL databases.

If your business allows you to stop all applications that access a PSQL database, you may use any of the operating system utilities, or third-party software, to backup or restore the database files.

Archival logging is another backup method that you can use to supplement operating system utilities. Archival logging allows you to keep a log of all database operations since your last backup. In case of a system failure, you can restore the data files from backup then roll forward the changes from the log file to return the database to the state it was in prior to the system failure.

Continuous operations allows you to backup database files while the database engine is running and users are connected. After starting Continuous Operations, the database engine closes the active data files and stores all changes in temporary data files (called delta files). When the backup is complete, you turn off Continuous Operations. The database engine then reads the delta file and applies all the changes to the original data files.

See Logging, Backup, and Restore for additional information about backing up and restoring databases.
Troubleshooting

The PSQL User's Guide and Getting Started With PSQL manuals both contain troubleshooting information. Getting Started With PSQL contains troubleshooting information pertaining to installing the PSQL products. The User's Guide contains general troubleshooting information as well as an extensive list of frequently asked questions (FAQs).

Other resources at your disposal for troubleshooting include the Actian Corporation Knowledge Base, which contains information based on actual customer solutions and common problems, and the Support area of the PSQL Web site.
Helpful Utilities

PSQL comes with a variety of utilities designed to help you control and manage your databases. For a list of the primary PSQL utilities, see PSQL User's Guide. Note that some utilities can be excluded if you perform a custom installation. See Getting Started With PSQL.
Concepts of Database Maintenance
A Discussion of Architecture Features

This chapter covers features designed to offer a trouble-free environment for installing and running critical applications. This chapter is divided into the following sections:

- PSQL Database Management System
- Relational Architectural Overview
- Error Codes
- PSQL Auto-Reconnect
The PSQL database management system consists of two database engines:
- MicroKernel Engine, which provides Btrieve API support for PSQL applications.
- Relational Engine, which provides ODBC support for PSQL applications.

Common Address Space
PSQL uses an optimized memory architecture that provides high performance for both transactional and relational data access methods. Both the MicroKernel Engine and the Relational Engine load and operate in the same process address space, minimizing the CPU time required to communicate between them.

Row Level Locking
Row level locking improves database engine performance in multiuser environments in which many updates and writes occur at the same time, or in which transactions remain open for an extended period of time.

A transaction locks only the rows that it affects directly, not the entire page. One client can update records on a given page at the same time as another client updates different records on the same page. Waiting is necessary only when a second application attempts to modify the exact same records currently locked by the first application. Thus, row level locking decreases overall wait time and improves performance in a multiuser environment.

This feature is completely transparent within the MicroKernel Engine. This feature is always on and is supported across Server and Workgroup products as well as all supported operating system platforms. This feature is supported for data file format v6.x and later.

Row level locking is implemented for data pages and partially implemented for key pages. Row level locking does not apply to variable pages. A small percentage of key page changes may cause key entries to move from one page to another. An example is when a key page is split or combined. These changes retain a full page lock until the transaction completes.

MicroKernel Engine
The MicroKernel Engine provides Btrieve API support for PSQL applications. There are two versions of the MicroKernel Engine. The Server Engine runs on Linux and Windows servers. The Workgroup Engine runs on Windows only and is designed for single-user or small workgroup configurations.

Both PSQL Server and PSQL Workgroup support local applications running on the same computer as the engine. The Server MicroKernel Engine supports both local applications and remote (client/server) applications. The Workgroup MicroKernel Engine supports applications running on remote machines as well and can service requests made by another peer Workgroup Engine on a remote machine.

The Workgroup Engine is by default configured to start up when you log into Windows. A Workgroup engine can service requests made by another peer engine if the files have already been opened by the engine. It can also serve in a gateway mode by configuring a particular machine and database engine to act as a gateway, thus preventing another Workgroup engine from opening the files.

The Server Engine for Windows is installed to run as a Windows Service. The Workgroup Engine can be installed to run as an application or as a service. By default, it is installed to run as a service for a fresh
install. If installed as an application, a “tray icon” is displayed to provide a graphical indication when a Workgroup Engine is running. No tray icon is displayed when the Workgroup Engine is not running. The tray icon does not display for the Server Engine or if the Workgroup Engine is installed as a service. See also Technical Differences Between Server and Workgroup.

The Btrieve and ODBC APIs in PSQL support writing distributed database applications that hide the details of connecting to a local or remote database engine from an application. Using this architecture, an application can access data that is co-located with the application (that is, running on the same computer as the application) while also accessing data on a remote computer. Moreover, a SQL database can be distributed by having DDFs (data dictionary files) serviced by a local MicroKernel Engine and data files (tables) serviced by a remote MicroKernel Engine. Such a SQL database, which is not serviced exclusively by a local MicroKernel Engine, is referred to as a “mixed access database.”

Mixed-access databases are subject to the following constraints:

- The following features are not supported: referential integrity (RI), bound databases, triggers, distributed transaction atomicity (requires two-phase commit).
- The Relational Engine and the MicroKernel Engine must be running on the same computer to access DDFs.
- Data files for tables that are involved in an RI relationship, or those that have any triggers defined for them, or are in a bound named database, cannot be opened by a remote MicroKernel Engine.
- When opening a file, the Relational Engine does not verify the version of the MicroKernel Engine servicing the request. If an operation that requires v6.30 or higher MicroKernel Engine API support (for example, shared locking) is issued to an engine less than v6.30, then an error code is returned. When opening DDFs or when attempting to bind a DDF or data file, the Relational Engine verifies that the local MicroKernel Engine is servicing the request.

Asynchronous I/O

The Server MicroKernel engine for Windows uses asynchronous I/O when writing pages to disk. This feature improves performance. The MicroKernel quickly writes pages to the Windows system cache or its own cache. In turn, Windows signals when the pages are on disk, helping the MicroKernel to perform write operations efficiently.

Read performance is also enhanced when there are many concurrent operations being done in the MicroKernel at the same time, especially if you access your data set on a striped set of disk drives. Each read causes a worker thread to wait until the page is available. With asynchronous I/O, the operating system can pool the work of multiple readers to make the read operations more efficient.

Relational Engine

The PSQL Relational Engine provides ODBC support for PSQL applications.

ODBC client platforms include Windows platforms. Remote ODBC application access to the Relational Engine requires installation of the ODBC Client, which is a specialized ODBC driver that routes client-side ODBC calls to the ODBC communications server over the network.

Some of the features of the Relational Engine include:

- Atomic statements
- Bidirectional cursors (using the ODBC Cursor Library)
- Outer join support
- Updatable views
The PSQL Component Architecture

- ODBC data type support
- Multiple variable length columns in a table

The ODBC communications server performs the following functions:

- supports network communication for ODBC Clients
- routes ODBC calls to the server-side ODBC Driver Manager (which, in turn, routes ODBC calls to the Relational Engine)

For additional details on SQL and ODBC, see SQL Overview in SQL Engine Reference and DSN Setup and Connection Strings ODBC Guide.
Relational Architectural Overview

The following diagram illustrates the architectural components of PSQL’s Relational Engine for the server version. The SQL Connection Manager starts and runs in the same process address space as the MicroKernel Engine and the Relational Engine.

**PSQL Relational Architecture: Server**

The SQL Connection Manager supports up to 2000 simultaneous connections and uses the ODBC Driver Manager to make calls to the Relational Engine (SRDE), which in turn rests on top of the MicroKernel.

Figure 2 illustrates the client/server relational architecture of PSQL. The client talks to the SQL Connection Manager on the server through TCP/IP. This architecture applies to the server engine and to the Workgroup engine (in the case where a client DSN is used to connect from the local Workgroup engine to the remote Workgroup engine).

![Client/Server Relational Architecture Diagram](image)

Figure 3 illustrates the Workgroup relational architecture when a DSN is used to connect from the local Workgroup engine to the remote database, assuming that a remote Workgroup engine is acting as a Gateway to the remote data.
Figure 3  Workgroup Relational Architecture
Error Codes

Most PSQL top-level components pass through error codes from underlying components so that the actual source of the error is clearly identified to the calling application or in the log file. In situations where an error code may apply to various situations, specific information in the PSQL event log should identify the root cause of the error. See Reviewing Message Logs.
**PSQL Auto-Reconnect**

PSQL Auto-Reconnect (PARC) allows client-server or workgroup applications to endure temporary network interruptions without canceling the current database operation. When PSQL detects a network interruption, it automatically attempts to reconnect at specific intervals for a configurable amount of time. This feature also preserves the client context so that when communications are re-established, database access continues exactly where it left off when the network interruption occurred.

This feature preserves the application context and attempts to reconnect regardless of whether the client or server was attempting to send data at the moment when the network communications were interrupted.

When a network interruption occurs, the reconnect attempts occur at specific intervals. For all connections, successive attempts are made at 0.5, 1, 2, 4, and 8 seconds, continuing every 8 seconds thereafter until the AutoReconnect Timeout value is reached. If no attempt is successful before the maximum wait time is reached, then the current operation fails and the client connection is reset. The maximum wait time is configurable between 45 seconds and 65,535 seconds.

This feature is disabled by default. For this feature to operate, you must select **Enable Auto Reconnect (Windows only)** for both client and server configurations. You can specify the time-out value using the server setting **Auto Reconnect Timeout**.

**Remarks**

This feature is supported for Btrieve, ODBC, and DTI connections.

The Btrieve communication servers may write out *.PAR* or *.SAR* files to the Transaction Log Directory. These are temporary files that contain the context for the last item that the server tried to send to the client. When a reconnection occurs, the client may ask for data to be re-sent. The server reads these files to obtain the appropriate data. These files are normally deleted by the server after the data is read or later when the connection is finally terminated.
This chapter discusses the following topics:

- Configuration Overview
- Configuration Through PCC
- Configuration Through CLI Utility
- Configuration Settings Parameters
- Services Configuration Parameters
- Server Configuration Parameters
- Windows Client Configuration Parameters
- Linux Client Configuration Parameters
Configuration Reference

Configuration Overview

Configuration is the process by which you provide settings for database engines and clients. You can specify configuration settings in PSQL Control Center (PCC) or with a command line interface (CLI) utility.

In PCC, the configuration settings are properties of the engine or client. See To access configuration settings in PCC for an engine and To access configuration settings in PCC for a local client.

Configuring any components is optional. If you do not configure them, each component loads with default configuration settings. For best results, you should use only the version of PCC that is the same version as your client and engine components.

You can use configuration for the following reasons:

- Your system or your PSQL application requires you to adjust the settings. Refer to your application documentation for recommended values. If you are running multiple applications concurrently, add the recommended values together. If you are running multiple applications sequentially, use the highest recommended value.
- You want to optimize the settings so that PSQL provides the services you need without using more memory than necessary. (The stated memory requirements provide guidelines for optimizing your computer's resources.)

The configuration settings themselves are discussed in Configuration Reference.

Ensuring Configuration Changes Take Effect

Some engine configuration settings require that you restart the database engines after the setting is changed. Restarting the engines ensures that the setting takes effect. Each setting in this chapter lists whether a database engine restart is required. Also, PCC informs you with a popup message if a changed setting requires an engine restart.

The CLI utility also informs you provided that you changed the setting from the command line rather by using an input file. Use of an input file always requires that you restart the engines. See Configuration Through CLI Utility.

To stop and start a server database engine from the command line, see the following topics in PSQL User’s Guide:

- Starting and Stopping the Server Engine on a Windows Server
- Starting and Stopping the Database Engine on Linux

To stop and start a Workgroup engine, see Starting and Stopping the Workgroup Engine on Windows in PSQL User's Guide.

In addition, changing client parameters often requires the client to be restarted. To re-load the client, simply exit all applications that depend on PSQL and restart them.

Connecting to Different Machines

You can configure both local and remote engines as well as local client components; however, each engine must be configured separately. See Configuration Through PCC and Configuration Through CLI Utility.
When you are connected to a remote machine with PCC, you can view and change only engine components. Client components (such as on workgroup and workstation engines and client machines) can only be configured locally on each machine.
Configuration Through PCC

In PCC, the configuration settings are properties of an engine or a client. All registered engines appear as nodes in PSQL Explorer and allow you to configure settings through the properties. Only the local client appears in PSQL Explorer.

You set configuration settings for clients at each client machine itself. By default, PCC is installed with the client components.

➢ To access configuration settings in PCC for an engine
1. In PSQL Explorer, expand the Engines node in the tree (click the expand icon to the left of the node).
2. Right-click the database engine for which you want to specify configuration settings.
3. Click Properties.
4. Click the desired option category in the tree to display the settings for that category of options.
5. Optionally, press shift+F1 to access help for the settings.

➢ To access configuration settings in PCC for a local client
1. In PSQL Explorer, expand the Local Client node in the tree (click the expand icon to the left of the node).
2. Right-click MicroKernel Router.
3. Click Properties.
4. Click the desired option category in the tree to display the settings for that category of options.
5. Optionally, press shift+F1 to access help for the settings.
Configuration Through CLI Utility

The command line interface (CLI) version of configuration provides the same configuration functionality as the property dialogs in PSQL Control Center. The CLI Configuration runs on the Windows and Linux platforms supported by PSQL v12.

On Windows, the executable program is `bcfg.bat` and is installed, by default, in the Program Files directory.

On Linux, the executable program name is `bcfg` and is located, by default, in the `/usr/local/psql/bin` directory. The following requirements must be met to run bcfg on Linux.

Table 6  Requirements for Running bcfg on Linux

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java Runtime Environment (JRE)</td>
<td>The JRE components required to run bcfg are installed as part of PSQL. Bcfg uses the “local” version of the JRE installed as part of PSQL.</td>
</tr>
<tr>
<td>PSQL server or client</td>
<td>A compatible PSQL server or client must already be installed on the same machine. See Installing PSQL Server, Vx Server and Client for Linux in Getting Started With PSQL.</td>
</tr>
</tbody>
</table>

If you have met the requirements to run bcfg on Linux and still are having difficulty running the utility, refer to the following troubleshooting guide.

Table 7  Troubleshooting Guide for Running bcfg on Linux

<table>
<thead>
<tr>
<th>Troubleshooting Condition</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>You receive the error &quot;java.lang.UnsatisfiedLinkError.&quot;</td>
<td>This error typically occurs if you try to start bcfg by double-clicking the script file using a file browser application. Start bcfg from a command prompt. This error can result if the LD_LIBRARY_PATH variable is not set. If you run bcfg as user &quot;psql,&quot; this variable is set in the profile for psql. You may also explicitly set the variable with the following command: <code>export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/psql/lib</code></td>
</tr>
<tr>
<td>You receive the following error message: &quot;Unable to connect to database engine. Make sure the target machine is accessible and an engine is running on the target machine.&quot;</td>
<td>The context of this error occurs if you attempt to administer the local server. To administer the local server, you must be a member of the psvw group or be the root user. See also PSQL Account Management on Linux in Getting Started With PSQL.</td>
</tr>
</tbody>
</table>

**Setting a Configuration**

You can configure settings one at a time from the command line or by providing one or more settings in an input file.
A convenient way to create an input file is first to create an output file. You can then edit the output file and use the edited version as an input file. See Editing an Input File for the types of edits permissible.

**Restarting the Engines**

If you use an input file, `bcfg` prompts you to restart the database engines after the utility processes the file. The restart applies regardless of the settings in the input file. Stopping then starting the engines ensures that the configuration settings take effect.

If you configure a setting from the command line, `bcfg` prompts you to restart the database engines only if the setting requires a restart.

See Ensuring Configuration Changes Take Effect for how to restart the engines.

**Example Scenario: Configuring a Single Setting from the Command Line**

Suppose that you want to turn on a configuration setting having to do with reconnecting a client to the server in the event of a network outage. You are not certain about the name of the configuration setting. Complete the following steps.

1. At a command prompt, type `bcfg -H reconnect` then press Enter.
   The utility reports all settings that contain the string “reconnect”:
   
<table>
<thead>
<tr>
<th>ID</th>
<th>Setting Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>29</td>
<td>Enable Auto Reconnect</td>
</tr>
<tr>
<td>148</td>
<td>Enable Auto Reconnect</td>
</tr>
<tr>
<td>149</td>
<td>Auto Reconnect Timeout</td>
</tr>
</tbody>
</table>

   Two of the settings, 29 and 148, look like what you want, but which is which?

2. Type `bcfg 29` then press Enter.
   The utility reports the following for setting ID 29:

   `=====================
   ENABLE AUTO RECONNECT
   ================`

   | ID: 29  Value: Off          |
   | Options: On Off             |
   | Default Value: Off          |
   | Description: \(<Client setting>\) Specifies if the Client will attempt to reconnect to the Server in the event of a network outage. The reconnected Client will continue processing as if no errors were encountered. |

   The description tells your that this setting applies to a client and that the setting is currently “off.”

3. Type `bcfg 29 on` then press Enter.
   The utility informs you that system setting 29 has been updated.

4. If you want to verify that the setting is now “on,” type `bcfg 29` then press Enter.
The utility reports the following for setting ID 29:

=====================  
ENABLE AUTO RECONNECT  
=====================  

ID: 29  
Value: On  
Options: On Off  
Default Value: Off  
Description: <Client setting> Specifies if the Client will attempt to reconnect to the Server in the event of a network outage. The reconnected Client will continue processing as if no errors were encountered.

Note that the value is now set to “on.”

Editing an Input File

An input file must contain at least one complete record for one setting. If you create an input file from an output file and want to remove configuration settings, ensure that the remaining settings are complete records.

At a minimum, a complete record encompasses an ID and Value pair. However, to ensure clarity, it is recommended that you include the top line of the setting header through the setting's description. For example, here is a suggested minimal record for Enable Auto Reconnect:

=====================  
ENABLE AUTO RECONNECT  
=====================  

ID: 29  
Value: On  
Options: On Off  
Default Value: Off  
Description: <Client setting> Specifies if the Client will attempt to reconnect to the Server in the event of a network outage. The reconnected Client will continue processing as if no errors were encountered.

Other than limiting which setting records are included in the input file, the only other change allowed is to the Value assignments. The assignment can be whatever is specified by Options or by Range.

Command Syntax

```
bcfg -I inputfile [-S server] [-U username]  
[-P password] [-E]  

or

bcfg -O outputfile [-S server] [-U username]  
[-P password] [-E]  

or

bcfg ID [value] [-S server] [-U username]  
[-P password]  
```

or
Configuration Reference

```
bcfg -H <keyword | ''keyword with spaces'' | [-S server] [-U username] [-P password]
```

**Options**

- **-I**
  Required parameter if you provide an input file to the utility.

  `inputfile`
  A text file that contains one or more configuration setting records for a specified server and the value assigned to each setting.

  A convenient way to create an input file is first to create an output file. You can then edit the setting values as required and use the edited version as an input file. See Editing an Input File for the types of edits permissible.

- **-O**
  Required parameter if you want the output results of running the utility sent to a text file.

  `outputfile`
  A text file that contains the current configuration settings for a specified server as a result of running the utility.

- **ID**
  A two or three digit integer that uniquely identifies the configuration setting.

  Some configuration settings require that you restart the database engines for the setting to take effect. Bcfg prompts you if a restart is required. See Restarting the Engines.

- **value**
  A value assigned to the configuration setting. The valid values are specified in a setting record in Options or Range.

  If Value is omitted, the utility returns the current setting.

  If Value is included, the utility changes the setting based on the value assignment.

  See Example Scenario: Configuring a Single Setting from the Command Line.

- **-H**
  The help search option for a `keyword` (that is, help for a `keyword`). The utility searches for configuration settings that contain `keyword` and returns the ID and setting name, or returns “no matches found.” See also next row in this table.

  `keyword`
  The name of the configuration setting, such as ALLOW CLIENT-STORED CREDENTIALS or SUPPORTED PROTOCOLS.

  Note that `keyword` is case insensitive. However, if the keyword contains one or more spaces, you must double quote the string.

  The utility can provide help based on partial keywords. For example, if you specify `-H client`, the utility returns all setting with the word “client” as part of the setting name. If you specify `-H a`, the utility returns all settings with an “a” in the name.

- **-S**
  Required parameter if the configuration settings apply to a remote server (a server other than the local one).

  `server`
  The name or IP address of the remote server that contains the database engine.

- **-U**
  Required parameter if a user name is required to access `server`.

  `username`
  User name with which you will connect to `server`. See also PSQL Security.

  If `server` is a local machine, the `username` and `password` are not required if:

  - You are logged in to the local machine as an administrator or as a member of the Pervasive_Admin group
  - The local machine is not running Terminal Services.

- **-P**
  Required parameter if a password is required to access `server`. 
password  Password used with username to connect to the server. See username. See also PSQL Security.

-E  Ignore errors when reading inputfile or writing to outputfile.
Configuration Settings Parameters

Each configuration setting lists a number of parameters in a tabular format.

<table>
<thead>
<tr>
<th>Column in Tabular Format</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the parameter.</td>
</tr>
</tbody>
</table>
| Type                     | Defines the value type. The different types are:  
  • Numeric  
  • Bool(ean)  
  • String  
  • Multisel(ect) – choose several values from a given list  
  • SelectOne – must choose one value from a given list |
| Value                    | Shows the current setting. |
| Units                    | If the value is numeric, this field explains any specific units of measure, such as KB or seconds, if required. |
## Services Configuration Parameters

On Windows server environments, PSQL Server runs as services. The services are loaded as part of the installation process and are set to be always available if you followed the Complete installation. You may configure the startup policy for the services from within PCC.

➢ **To set services startup policy by using PCC**

1. Access Control Center from the operating system Start menu or Apps screen.
2. In the PSQL Explorer, expand the Services node (click the expand icon to the left of the node to display the subordinate nodes).
3. Right-click the desired service: PSQL (relational) or PSQL (transactional).
4. Click Properties.
5. Click the desired startup policy:

<table>
<thead>
<tr>
<th>Policy</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manual</td>
<td>The service is not started automatically when the operating system starts. You must manually start the service after the operating system starts or restarts.</td>
</tr>
<tr>
<td>Automatic</td>
<td>The service is started automatically when the operating system starts or restarts.</td>
</tr>
<tr>
<td>Disabled</td>
<td>The service is rendered inoperative until you re-set the startup policy to Manual or to Automatic.</td>
</tr>
</tbody>
</table>

6. Click **OK**.
Server Configuration Parameters

Each PSQL database engine has its own server configuration options. This section describes the different configuration options available for the engines.

You can configure PSQL Servers on Windows and Linux platforms using the graphical utility PSQL Control Center. You can also use the command-line interface utility `bcfg`. For PCC, see Using PSQL Control Center in PSQL User’s Guide. For `bcfg`, see Configuration Through CLI Utility.

The following table lists the Server configuration options and their settings.

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Setting Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Accept Remote Request</td>
</tr>
<tr>
<td></td>
<td>Allow Cache Engine Connections</td>
</tr>
<tr>
<td></td>
<td>Allow Client-stored Credentials</td>
</tr>
<tr>
<td></td>
<td>Authentication (Linux engines only)</td>
</tr>
<tr>
<td></td>
<td>Configuration File (Linux engines only)</td>
</tr>
<tr>
<td></td>
<td>Prompt for Client Credentials</td>
</tr>
<tr>
<td></td>
<td>Wire Encryption</td>
</tr>
<tr>
<td></td>
<td>Wire Encryption Level</td>
</tr>
<tr>
<td>Communication Protocols</td>
<td>Auto Reconnect Timeout</td>
</tr>
<tr>
<td></td>
<td>Enable Auto Reconnect (Windows only)</td>
</tr>
<tr>
<td></td>
<td>Listen IP Address</td>
</tr>
<tr>
<td></td>
<td>NetBIOS Port (Workgroup engines only)</td>
</tr>
<tr>
<td></td>
<td>Supported Protocols</td>
</tr>
<tr>
<td></td>
<td>TCP/IP Multihomed</td>
</tr>
<tr>
<td></td>
<td>TCP/IP Port</td>
</tr>
<tr>
<td>Compatibility</td>
<td>Create File Version</td>
</tr>
<tr>
<td></td>
<td>System Data</td>
</tr>
<tr>
<td>Data Integrity</td>
<td>Archival Logging Selected Files</td>
</tr>
<tr>
<td></td>
<td>Initiation Time Limit</td>
</tr>
<tr>
<td></td>
<td>Operation Bundle Limit</td>
</tr>
<tr>
<td></td>
<td>Transaction Durability</td>
</tr>
<tr>
<td></td>
<td>Transaction Logging</td>
</tr>
<tr>
<td></td>
<td>Wait Lock Timeout</td>
</tr>
</tbody>
</table>
## Server Configuration Parameters

### Access

Access contains the following configuration settings:

- Accept Remote Request
- Allow Cache Engine Connections
- Allow Client-stored Credentials
- Authentication (Linux engines only)

### Table 9  Server Configuration Options and Settings

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Setting Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Debugging</td>
<td>Number of Bytes from Data Buffer</td>
</tr>
<tr>
<td></td>
<td>Number of Bytes from Key Buffer</td>
</tr>
<tr>
<td></td>
<td>Select Operations</td>
</tr>
<tr>
<td></td>
<td>Trace File Location</td>
</tr>
<tr>
<td></td>
<td>Trace Operation</td>
</tr>
<tr>
<td>Directories</td>
<td>DBNames Configuration Location</td>
</tr>
<tr>
<td></td>
<td>Transaction Log Directory</td>
</tr>
<tr>
<td></td>
<td>Working Directory</td>
</tr>
<tr>
<td>Information</td>
<td>Server name (display-only)</td>
</tr>
<tr>
<td></td>
<td>Engine version (display-only)</td>
</tr>
<tr>
<td></td>
<td>Engine type (display-only)</td>
</tr>
<tr>
<td>Memory Usage</td>
<td>Allocate Resources at Startup</td>
</tr>
<tr>
<td></td>
<td>Back to Minimal State if Inactive</td>
</tr>
<tr>
<td></td>
<td>Minimal State Delay</td>
</tr>
<tr>
<td></td>
<td>Sort Buffer Size</td>
</tr>
<tr>
<td></td>
<td>System Cache</td>
</tr>
<tr>
<td>Performance Tuning</td>
<td>Cache Allocation Size</td>
</tr>
<tr>
<td></td>
<td>Communications Threads</td>
</tr>
<tr>
<td></td>
<td>File Growth Factor</td>
</tr>
<tr>
<td></td>
<td>Index Balancing</td>
</tr>
<tr>
<td></td>
<td>Limit Segment Size to 2 GB</td>
</tr>
<tr>
<td></td>
<td>Log Buffer Size</td>
</tr>
<tr>
<td></td>
<td>Max MicroKernel Memory Usage</td>
</tr>
<tr>
<td></td>
<td>Number of Input/Output Threads</td>
</tr>
<tr>
<td></td>
<td>Transaction Log Size</td>
</tr>
</tbody>
</table>

__Access__

Access contains the following configuration settings:
- **Configuration File (Linux engines only)**
- **Prompt for Client Credentials**
- **Wire Encryption**
- **Wire Encryption Level**

### Accept Remote Request

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>On</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies whether the Communications Manager accepts requests from remote servers and client workstations. If you turn this option to **On**, the Communications Manager advertises its presence on the network.

### Allow Cache Engine Connections

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>On</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Specifies if the server will support clients that will attempt to connect to the server with the Cache Engine. When set to **Off**, clients will still connect to the Server but will not use the Cache Engine.

### Allow Client-stored Credentials

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>On</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

When this setting is **On**, the database engine accepts user credentials stored on the client. The method and location of storage depends on the operating system of the client:

- **Windows clients**: these credentials are stored in the Windows registry. When the **Prompt for Client Credentials** is set to **On**, then a pop-up dialog allows you to save the credentials by selecting the **Save User name and Password** check box. Alternatively, you can use the `pvnetpass` command-line utility to manage stored credentials.
- **Linux clients**: credentials are stored in the PSQL registry by the `pvnetpass` utility.

When this setting is **Off**, the database engine forces the client to omit stored credentials from any database operation that requires credentials. Such credentials must be supplied by the application or through the login dialog. The login dialog still writes client-stored credentials if specified using the login dialog, even if this setting is Off. However, they will not be accepted.

When client-stored credentials are allowed, anyone can sit at that particular client computer and log in to the database using the stored credentials without knowing those credentials. This behavior can be convenient for environments in which strict authentication of individual users is not a concern, such as...
a physically secured work area where all users have the same level of access permissions. On the other hand, in environments where unauthorized personnel are present or authorized users have varying levels of access permissions, this setting must be **Off**.

See also: Prompt for Client Credentials.

**Summary Chart of Login Behavior**

<table>
<thead>
<tr>
<th>Prompt for Credentials</th>
<th>Allow Client-Stored Credentials</th>
<th>Behavior</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>Off</td>
<td>PSQL client does not prompt the user or use stored credentials, thus credentials must be supplied by the client application during a Btrieve operation.</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>If credentials are not supplied by the client application during the Btrieve operation, the client uses credentials stored by the login dialog or by pvnetpass, if such credentials are available. If no credentials are supplied by either method, the connection attempt fails. No login dialog is displayed.</td>
</tr>
<tr>
<td>On</td>
<td>Off</td>
<td>If credentials are not supplied by the client application during the Btrieve operation, the client displays a login dialog to the user, and the Linux client returns a status code for permissions error. Credentials stored by the login dialog or by pvnetpass are not used.</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>If credentials are not supplied by the client application during the Btrieve operation, stored credentials are used. If no stored credentials are available, then the client displays a login dialog to the user, and the Linux client returns a status code for permissions error.</td>
</tr>
</tbody>
</table>

**Authentication (Linux engines only)**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectOne</td>
<td>Three options. See below</td>
<td>Emulate Workgroup Engine</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The following options specify which type of authentication to use for access to the server engine. The available options are:

- **Emulate Workgroup Engine.** Use this value when Samba is used to authenticate user access on the system. If you want to bypass security provided by the operating system and do not want to store RTSS passwords in the registry, use **Emulate Workgroup Engine**.

- **Proprietary Authentication (using btpasswd).** Use this value when not using Samba to authenticate and the user does not have an account on the server. This allows a separate password file to be maintained when connecting to the Linux system.

If you are using BTPASSWD authentication on your Linux server, user names and passwords must be set from clients connecting to this server. Use PSQL Control Center or the pvnetpass utility. See Groups, Users, and Security and pvnetpass, both topics in the PSQL User's Guide.

Use Proprietary Authentication if stronger security is needed for the server and you want user names and passwords different from any user authentication scheme employed on the Linux server.
- **Standard Linux Authentication.** Use this value when not using Samba to authenticate but users have accounts on the Linux system.

  Standard Linux authentication is used with PAM. Use PAM if you want to use existing user names and passwords on the Linux server. You can specify user names and passwords from the client using the `pvnetpass` utility. PAM is also very flexible and offers many custom modules for Linux. Check the PAM home page on the Web for more information.

  If the PSQL installation detects PAM, the installation completes its configuration so that PAM can be used. If you install PAM after installing PSQL and want to use standard authentication with PAM, you must re-install PSQL. The reason is that the PAM installation copies files, creates configuration files, sets permissions, and creates links. PSQL needs to be re-installed to detect PAM and correctly complete its PAM configuration.

  You re-install PSQL by uninstalling and then installing again. See the chapter on **Installing PSQL Server, Vx Server and Client for Linux** in *Getting Started With PSQL* for the steps to uninstall and install.

**Samba and Authentication**

You may use Samba, if available, in addition to any of the three authentication methods described above. The server creates a well-known FIFO share via Samba. FIFO is created in `$PVSW_ROOT/etc/pipe/mkde.pip`. `$PVSW_ROOT/etc/pipe` should be shared by Samba as `PVPIPE$`.

- **Note** The trailing `$` means this share will be hidden. The PSQL client components automatically take care of accessing this pipe as `\<server>\PVPIPE$\mkde.pip` (case-insensitive); you do not need to perform any explicit actions or modify your application to access this pipe. The only exception to this is if you are troubleshooting your Samba or PSQL configurations (see section on Troubleshooting below).

When a client connects to the remote engine and discovers the engine returns UNIX in the version block, it will first look in the registry (RTSS) setting) for authentication information. If the user name and password are not found there, the client connects to the above pipe and receives client authentication information from the server, which will be validated later.

To be authenticated, you must be able to connect to the share and read the pipe. This is one way of specifying who can use the engine and who cannot. The easiest way to do this is to utilize the Samba “valid users” setting in `smb.conf` (Samba configuration file). If the client is unable to get authentication, status 3119 is returned.

The installation program sets up the Samba share (if Samba is installed on the server). For reference, here is an example of setting up the PSQL pipe.

```
[PVPIPE$]
comment = PSQL pipes
path = /usr/local/psql/etc/pipe
force group = pvsw
  # force group pvsw when accessing pipe - will be
  # useful if primary group for this user is not pvsw
valid users = @pvsw
  # only members of group pvsw will have access
```
oplocks = False

# Absolutely necessary - prevents caching on the client

To configure access to files shared through Samba, read the Samba documentation.

**Note** By allowing a client read access to PVPIPE$, that client is authorized to access the engine remotely.

A simple way to ensure the client gets proper authentication is to enter \\<yourserver>\pvpipe$\mkde.pip at the command prompt. You should see a lot of question marks (unprintable symbols), occasional printable characters and hear beeps. If you do not, check your Samba configuration to be sure you have rights to read this pipe. If you do but still get error 94 or 3119, validate your RTSS setting using the engine configuration properties in PSQL Control Center or with pvnetpass.

### Configuration File (Linux engines only)

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>not applicable</td>
<td>/etc/smb.conf</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the location of the smb.conf file used by Linux to export local file systems to Windows clients. The engine requires this file to translate UNC paths on remote systems into local calls to the correct database file.

The default value is /etc/smb.conf. If you installed the Samba configuration file in a different location, enter the correct path and/or file name.

### Prompt for Client Credentials

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

This setting determines whether the Windows PSQL client prompts the user for login credentials if no other credentials are available during a database operation that requires user authentication.

When this setting is **On**, in the absence of other authentication credentials, the engine requires the Windows client to present a login dialog to the user. This setting only applies when Mixed or Database security is in effect, and does not apply to the Linux client under any circumstances. If valid credentials are supplied via another method (for example, explicit Btrieve Login (78) operation or credentials stored on the client), the login dialog does not appear.

If no database context is specified to the engine within the operation requiring user credentials, the engine assumes the user is attempting to log in to the current database.

When this setting is **Off** and one of the new security models is in use, user credentials must be provided programmatically (credentials stored on the client or provided with a Btrieve Login (78), Open (0), or Create (14) operation), or else the login attempt fails with an authentication error.
See Also
Allow Client-stored Credentials

Wire Encryption

This parameter specifies whether the given client or server should use encryption for its network communications. The default value of If Needed means that the client or server only uses encryption if the other end of the communication stream requires it. For example, assume that Server A has its Wire Encryption value set to Always. Server B has its value set to Never. Your client has its value set to If Needed. In this case, the client will use encryption when communicating with Server A, but it will not use encryption when communicating with Server B.

The following chart summarizes the behavior given each possible combination of client and server values:

Table 11  Client/Server Results of Wire Encryption Combinations

<table>
<thead>
<tr>
<th>Client Setting</th>
<th>Server Setting “Never”</th>
<th>Server Setting “Always”</th>
<th>Server Setting “If Needed”</th>
</tr>
</thead>
<tbody>
<tr>
<td>Never</td>
<td>Encryption not used</td>
<td>Status Code 5001</td>
<td>Encryption not used</td>
</tr>
<tr>
<td>Always</td>
<td>Status Code 5000</td>
<td>Encryption used; level determined by highest Wire Encryption Level setting between client and server</td>
<td>Encryption used; level determined by client’s Wire Encryption Level setting.</td>
</tr>
<tr>
<td>If Needed</td>
<td>Encryption not used</td>
<td>Encryption used; level determined by server’s Wire Encryption Level setting</td>
<td>Encryption not used</td>
</tr>
</tbody>
</table>

Wire Encryption Level

This setting specifies the strength of the encryption key that should be used for encrypted communications. The following levels are available:

Table 12  Meaning of Encryption Level Values

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low</td>
<td>40-bit encryption key used</td>
</tr>
</tbody>
</table>
Server Configuration Parameters

Encryption using a key 128 bits long is generally accepted as “strong” encryption. The other settings provide progressively less protection but higher performance, in the event that you require some level of encryption but are willing to accept a lower level of deterrence to gain better performance.

When a client and a server both require encryption and one specifies a stronger encryption level than the other, the two entities use the stronger level to communicate.

Communication Protocols

Communication Protocols contains the following configuration settings:

- **Auto Reconnect Timeout**
- **Enable Auto Reconnect (Windows only)**
- **Listen IP Address**
- **NetBIOS Port (Workgroup engines only)**
- **Supported Protocols**
- **TCP/IP Multihomed**
- **TCP/IP Port**

**Auto Reconnect Timeout**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>45 - 65535</td>
<td>180</td>
<td>seconds</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies how long the client will attempt to connect to the server before giving up. When a AutoReconnect-enabled client first connects to a AutoReconnect-enabled server, the server communicates this value to the client so that both components know how long to attempt to reconnect in the event of a network interruption.

**Enable Auto Reconnect (Windows only)**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On Off</td>
<td>Off</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies whether you want the server to support clients attempting to auto-reconnect during a network outage. A setting of **On** means AutoReconnect is enabled.
Auto Reconnect is not in effect for a given client connection unless this setting is also enabled in that client's configuration.

To specify how long a client will attempt to reconnect to the server before giving up, see Auto Reconnect Timeout, above.

**Listen IP Address**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Valid IP address or multiple addresses separated by a comma between each address</td>
<td>0.0.0.0</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This option specifies the IP address or addresses the database engine listens on when TCP/IP Multihomed is Off. This option is ignored when TCP/IP Multihomed is On.

Multiple IP addresses may be specified but must be separated by a comma between each address. The string can be a combination of IPv4 and IPv6 addresses. Any of the IPv6 address formats supported by PSQL can be used. See Drive-based Formats in Getting Started With PSQL.

**NetBIOS Port (Workgroup engines only)**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>33 to 254</td>
<td>66</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This option specifies the NetBIOS port the MicroKernel listens on. The Server engine does not support NetBIOS.

**Supported Protocols**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiple select</td>
<td>See below</td>
<td>TCP/IP</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the protocols on which the database engine listens for client connections. If more than one protocol is specified, the database engine listens on all specified protocols. The default is TCP/IP. The available options are:

- TCP/IP
- SPXII
- NetBIOS

You must have at least one protocol enabled at both the client and the server or they cannot communicate.

*PSQL Workgroup*

NetBIOS is valid only for PSQL Workgroup, not PSQL Server.
**Linux**

TCP/IP is the only supported protocol for PSQL running on a Linux platform. Therefore, the Supported Protocols setting is not available on Linux.

**TCP/IP Multihomed**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>On</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This option specifies whether the database engine should listen for client connections on all network interfaces. If it is set to **On**, the database engine listens on all network interfaces, and the IP address(es) listed in the *Listen IP Address* option is ignored. If this setting is **Off**, you must specify in *Listen IP Address* which address(es) for the database engine to use for client communications.

**TCP/IP Port**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>256 to 65535</td>
<td>1583</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting configures the port number used by the Relational Engine.

This port number must be the same as that defined in any Client DSNs pointing to this server. For information on how to change the port number in a Client DSN, see Advanced Connection Attributes in ODBC Guide.

For additional information on ports, see Changing the Default Communication Ports in Getting Started With PSQL.

**Compatibility**

Compatibility contains the following configuration settings:

- Create File Version
- System Data

**Create File Version**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>SelectOne</td>
<td>6.x - 9.x</td>
<td>9.x</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

This setting specifies the format in which all new files are created. The 10.x database engine can write to files using the existing file format. In other words, it writes to 7.x files using the 7.x file format, writes to 8.x files using the 8.x file format, and so forth. (The 10.x database engine can read files created with 5.x, 6.x, 7.x, 8.x, and 9.x versions of the database engine.)

Specify 6.x, 7.x, or 8.x only if you need backward compatibility with a previous version of the MicroKernel. Specifying a previous file version does not affect any existing 9.x files.
**Note** Dictionary files (DDFs) must be created with a file format of 6.x or later. The **New Database** wizard uses the setting for create file version. The data files can be in any of the previous file formats supported. Only the DDFs must use a file format of 6.x or later.

### System Data

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single select</td>
<td>See below</td>
<td>If needed</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

System data refers to a hidden unique key in each record. Because the MicroKernel relies on uniquely identifying rows in order to ensure transaction durability, a file must either have a unique key defined or have system data included in the file. The default value is **If needed**; the available options are:

- **None**. By default, system data is not included on file creation. Application developers using the Create operation can override this setting.
- **If needed**. System data is added to the file on file creation if the file does not have a unique key.
- **Always**. System data is always added on file creation, regardless of whether the file has a unique key.

**Note** The System Data setting does not affect existing files. This setting only affects how new files are created.

If you want to use transaction durability with a file that was not created with System Data turned on and does not have a unique key, you must re-create the file after setting System Data to **Yes** or **If needed**.

The Relational Engine always creates files with System Data. This information applies to files created through SQL, OLE-DB, JDBC, or any method other than the Btrieve API.

Even if a file has a unique key, you may want to include system data, because users can drop indexes.

### Data Integrity

Data Integrity contains the following configuration settings:

- **Archival Logging Selected Files**
- **Initiation Time Limit**
- **Operation Bundle Limit**
- **Transaction Durability**
- **Transaction Logging**
- **Wait Lock Timeout**
Archival Logging Selected Files

This setting controls whether the MicroKernel performs archival logging, which can facilitate your file backup activities. If a system failure occurs, you can use the archival log files and the `BUTIL-ROLLFWD` command to recover changes made to a file between the time of the last backup and a system failure.

To direct the MicroKernel to perform archival logging, you must specify the files for which the MicroKernel is to perform archival logging by adding entries to an archival log configuration file that you create on the volume that contains the files. For more information about archival logging, refer to Understanding Archival Logging and Continuous Operations.

Initiation Time Limit

This setting specifies the time limit that triggers a system transaction. The MicroKernel initiates a system transaction when it reaches the `Operation Bundle Limit` or the time limit, whichever comes first, or when it needs to reuse cache.

Operation Bundle Limit

This option specifies the maximum number of operations (performed on any one file) required to trigger a system transaction. The MicroKernel initiates a system transaction when it reaches the bundle limit or the `Initiation Time Limit`, whichever comes first, or when it needs to reuse cache.

The MicroKernel Database Engine treats each user transaction (starting with Begin Transaction until End Transaction or Abort Transaction) as one operation. For example, if there are 100 Btrieve operations between the Begin Transaction and the End Transaction operation, then all the 102 Btrieve operations together are treated as a single operation.

Transaction Durability

Transaction Durability is the same as Transaction Logging except that Transaction Durability guarantees that all successfully completed transactions are committed to the data files in the event of a system crash.
For a full discussion of transaction logging and durability, see Transaction Logging and Durability.

**Note** When you turn Transaction Durability on, some files may not be able to support the feature. A file must contain at least one unique key, or when it was created, the configuration setting System Data must have been set to “Yes” or “If Needed.” Otherwise, any changes to the file are not written to the transaction log. For more information about transaction durability and system data, see PSQL Programmer's Guide.

Because System Data does not affect existing files, you may need to recreate files that do not have a unique key and were not created with System Data turned on. Be sure to turn on System Data before recreating these files.

**Caution** Gateway locator files allow different engines to manage files in different directories on the same file server. If your database contains data files in different directories, you must be sure that the same database engine manages all the data files in the database. If you have more than one database engine managing files within the same database, database integrity and transaction atomicity are not guaranteed. For more information on how to avoid this potential problem, see Redirecting Locator Files.

**Related Settings**
See more information on similar and related settings under Transaction Logging.

**Transaction Logging**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On Off</td>
<td>On</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting controls whether the MicroKernel ensures atomicity of transactions by logging all operations that affect the data files.

If the related setting, Transaction Durability, is turned on, then logging takes place automatically, and the Transaction Logging setting is ignored.

For a full discussion of transaction logging and durability, see Transaction Logging and Durability.
**Note** When you turn Transaction Logging on, some files may not be able to support the feature. A file must contain at least one unique key, or when it was created, the configuration setting **System Data** must have been set to “Yes” or “If Needed.” Otherwise, any changes to the file are not written to the transaction log. For more information about transaction durability and system data, see PSQL Programmer’s Guide in the Developer Reference.

Because System Data does not affect existing files, you may need to re-create files that do not have a unique key and were not created with System Data turned on. Be sure to turn on System Data before re-creating these files.

**Caution** Do not turn off Transaction Logging unless your database does not require transaction atomicity among data files. Database integrity for multifile databases cannot be guaranteed if Transaction Logging is turned off.

Do not turn off Transaction Logging unless doing so is supported by your application vendor.

**Related Settings**

The server configuration setting **Transaction Durability** is similar to Transaction Logging, but provides a higher level of data safety along with a lower level of performance. The server configuration settings **Log Buffer Size** and **Transaction Log Size** are related to **Transaction Logging**. **Log Buffer Size** allows you to configure the balance between transaction recoverability and performance. The larger the log buffer, the fewer times it is written to disk, and thus the greater the performance. However, database changes that are in the log buffer are not durable through a system failure.

**Transaction Log Size** controls how large each log file segment gets before a new segment is started.

Note that all of these settings are ignored if Btrieve or SQL transactions are not being used.

**Wait Lock Timeout**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0 - 2147483647</td>
<td>30000</td>
<td>milliseconds</td>
<td>Yes</td>
</tr>
</tbody>
</table>

The database engine and its clients use a coordinated retry mechanism when a record lock conflict occurs. If the engine cannot obtain a lock on every requested record within the duration for wait lock timeout, the engine returns control to the application with an appropriate status code.

**Wait Lock Timeout Benefits**

Wait lock timeout provides the following benefits if a lock conflict occurs:

- Allows the database engine to continue processing requests while waiting for the lock to release.
- Improves thread queuing if multiple threads are waiting for the locked resource.
- Improves network performance by reducing network traffic.
When Wait Lock Timeouts Apply

Wait lock timeouts apply to only two kinds of applications:

- Any application that uses the Relational Engine
- Btrieve applications performing a change operation that does not need to be retried. Such applications receive a lock error within either one second or the wait lock timeout value, whichever is less.

Wait lock timeouts do not usually apply to Btrieve applications that use the MicroKernel Engine through a PSQL client on Windows or Linux. Instead, such applications do one of the following:

- Receive a page or lock conflict error immediately for read operations with “no wait” lock bias (200, 400) or a write operation to which a “no write wait” lock bias (500) has been applied.
- Receive a page or lock conflict error within the lesser of either 1 second or the wait lock timeout value for nontransactional write operations where a “no write wait” lock bias (500) has not been applied.
- Wait indefinitely if the operation involved is a read operation with a “wait” lock bias (100, 300).
- Wait indefinitely if the operation involved is a write operation inside of a transaction and a “no write wait” lock bias (500) has not been applied to either the operation or the transaction.

On receiving a page or lock conflict error, the application may determine how to handle the conflict by retrying, waiting, or other options.

Handling of Page Locks

The MicroKernel Engine API provides controls to handle record lock situations. See Btrieve API Guide in the PSQL SDK documentation for a complete discussion. In brief, here are the control mechanisms:

- Explicit record locks - You can add lock biases (100, 200, 300, or 400) to read operations when reading records to specify whether to wait. You may also apply these biases to the Begin Transaction operation.
- Implicit page locks during a transaction - Most page lock conflicts are avoided because of row-level locking, but you can add lock bias 500 to your transactions to avoid waiting when a page lock occurs.
- Exclusive file locks - Use concurrent transactions to avoid explicit file locks. If a file cannot be opened exclusively, the request always returns immediately.

Debugging

Debugging contains the following configuration settings:

- Number of Bytes from Data Buffer
- Number of Bytes from Key Buffer
- Select Operations
- Trace File Location
- Trace Operation
Server Configuration Parameters

Number of Bytes from Data Buffer

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0 - 65535</td>
<td>128 bytes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

This setting specifies the size of the data buffer that the MicroKernel writes to the trace file. The Trace Operation setting must be set to On to use this setting. The size you specify depends on the nature of your tracing needs (whether you need to see the entire data buffer contents or just enough of the buffer contents to identify a record).

Number of Bytes from Key Buffer

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0 - 255</td>
<td>128 bytes</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

This setting specifies the size of the key buffer that the MicroKernel writes to the trace file. The Trace Operation setting must be set to On to use this setting. The size you specify depends on the nature of your tracing needs (whether you need to see the entire key buffer contents or just enough of the buffer contents to identify a key).

Select Operations

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multiselect</td>
<td>See below</td>
<td>All</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

The Selected list displays the available Btrieve API operation codes that are traced. Select from the list the desired operations to trace.

- Abort Transaction (21)
- Begin Transaction (19)
- Clear Owner (30)
- Close (1)
- Create (14)
- Create Index (31)
- Delete (4)
- Drop Index (32)
- End Transaction (20)
- Extend (16)
- Find Percent (45)
- Get By Percent (44)
- Get Position (22)
- Get Previous (7)
- Get Previous Extended (37)
- Insert (2)
- Insert Extended (40)
- Open (0)
- Reset (28)
- Set Directory (17)
- Set Owner (29)
- Stat (15)
- Step First (33)
- Step Last (34)
### Trace File Location

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>not applicable</td>
<td>file_path\PSQL\bin\mkde.tra</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

This setting specifies the trace file to which the MicroKernel writes trace information. The file name must include a drive or volume specification and path or use a UNC path. If you do not want the trace file in the default location, enter a different path and/or file name.

For default locations of PSQL files, see Where are the PSQL files installed? in Getting Started With PSQL.

**Note** Do not use the same trace file name for ODBC tracing and MicroKernel tracing.

### Trace Operation

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On/Off</td>
<td>Off</td>
<td>None</td>
<td>No</td>
</tr>
</tbody>
</table>

This setting enables or disables the trace feature, which allows you to trace each Btrieve API call and save the results to a file. Developers can use tracing to debug applications. The MicroKernel writes to the trace file using forced write mode, which ensures that data gets written to the file even if the MicroKernel unloads abnormally. The MicroKernel’s performance can be severely impacted, depending on the frequency of incoming requests. If you enable this option, you must specify a Trace File.
**Note** You do not need to restart the engine in order to start and stop tracing. You can turn tracing on or off during runtime and apply the changes directly to the engine. If you receive a message from Configuration indicating that you must restart the engine for changes to take effect, you may safely ignore the message for this setting.

**Directories**

Directories contains the following configuration settings:

- **DBNames Configuration Location**
- **Transaction Log Directory**
- **Working Directory**

**DBNames Configuration Location**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>not applicable</td>
<td>Varies</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the path to an alternate location for the DBNames configuration file. For Server engines, this is a local file path, not a directory path. For Workgroup engines this could be a remote path that is accessible to the Workgroup MicroKernel. The defaults vary depending upon your particular engine platform.

- Windows platforms: `application_data_directory\`
- Linux server: `/usr/local/psql/etc`

If you do not want the configuration file in the default location, enter a valid path.

**Transaction Log Directory**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>not applicable</td>
<td>Varies</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the location the MicroKernel uses to store the transaction log. The path must be a valid path and include a drive or volume specification or UNC path. The defaults vary depending upon your operating system.

The engine ignores this setting unless **Transaction Durability** or **Transaction Logging** is turned on.

**Caution** Do not use the same directory for multiple database engines (for example, specifying a remote server directory as the Transaction Log Directory for more than one Workgroup engine). Under these circumstances, the engines cannot determine which transaction log segments are used by each engine in the event a log roll forward is necessary.
Configuration Reference

If your database engine is highly utilized, you should configure your system to maintain the transaction logs on a separate physical volume from the volume where the data files are located. Under heavy load, performance is typically better when the log writes and data file writes are split across different drives instead of competing for I/O bandwidth on a single drive. For a full discussion of transaction logging, see Transaction Logging and Durability.

**Working Directory**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>not applicable</td>
<td>Same directory as data file</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the location of the MicroKernel working directory, which is used to store temporary files in operations such as building large indexes. If disk space is limited on certain volumes, you can use this option to specify a working directory on a volume with adequate space.

There is no default value specified; however, if you do not specify a working directory, the default will be the location of the data file. To specify a fixed working directory, enter a path in the Value text box. The path must include a drive or volume specification or a UNC path.

**Information**

Information lists the following display-only items:

<table>
<thead>
<tr>
<th>Display-only Item</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server name</td>
<td>The name of the machine on which the database engine is running.</td>
</tr>
<tr>
<td>Engine version</td>
<td>The release version of the database engine.</td>
</tr>
<tr>
<td>Engine type</td>
<td>The product category of the database engine, such as Server or Workgroup.</td>
</tr>
</tbody>
</table>

**Memory Usage**

Memory Usage contains the following configuration settings:

- Allocate Resources at Startup
- Back to Minimal State if Inactive
- Minimal State Delay
- Sort Buffer Size
- System Cache

**Allocate Resources at Startup**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On/Off</td>
<td>Off</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting instructs the MicroKernel to allocate resources, including threads and memory buffers, when the MicroKernel is started. The resources allocated at startup includes the background threads in
addition to the L1 cache. PSQL components automatically allocate resources as needed. Therefore, in most cases, this setting can be “off” (the default).

With the setting “off,” the MicroKernel does not allocate any resources until the first operation request. If your server system supports a large number of users, you may prefer to have this setting “on.”

When first set to “on,” the setting may not produce any noticeable difference in the memory allocated because of how Windows behaves. When the MicroKernel allocates its L1 cache, the Windows operating system simply reserves pages of memory and does not actually commit them to the MicroKernel. Later, when the MicroKernel actually accesses cache memory, Windows then commits actual physical pages and the memory usage of PSQL components (such as ndbsmgr or w3dbsmgr) increases.

If you look at the “VM Size” column in Windows Task Manager, you can see the memory value changing when the L1 cache gets accessed. You should also be able to see a difference in the number of threads when the background threads are accessed.

**Back to Minimal State if Inactive**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting causes the MicroKernel to free considerable memory and thread resources to the system and return to a minimal state after a certain amount of time without any active clients. The time interval is specified by the value of **Minimal State Delay**. The MicroKernel reallocates resources when another client becomes active.

**Minimal State Delay**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0 - 2147483647</td>
<td>300 milliseconds</td>
<td>Yes</td>
<td></td>
</tr>
</tbody>
</table>

This setting specifies how long the MicroKernel waits during a period of inactivity before returning to a minimal state. (This is the initial state in which the MicroKernel begins.) By returning to a minimal state, the MicroKernel frees considerable memory and thread resources to the system. In some cases, you may not want the MicroKernel to return to a minimal state. For example, you may be running a batch file that uses the MicroKernel repeatedly. The MicroKernel reallocates resources when another client becomes active.

This setting is ignored if the value of **Back to Minimal State if Inactive** is set to **Off** (the default).

**Sort Buffer Size**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0 - limited by memory</td>
<td>0</td>
<td>bytes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the maximum amount of memory (in kilobytes) that the MicroKernel dynamically allocates and de-allocates for sorting purposes during run-time creation of indexes.
If the memory required for sorting exceeds the size specified or is greater than 60 percent of the available process memory, the MicroKernel creates a temporary file. The amount of available memory for a process is a dynamic value and varies according to system configuration and load. If you specify 0 kilobytes, the MicroKernel allocates as much memory as needed, up to 60 percent of the available memory.

**System Cache**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On, Off</td>
<td>Off (Server engine)</td>
<td>None</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>On (Workgroup engine)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

This option specifies whether the MicroKernel should use the system cache in addition to the MicroKernel's own database cache, as set using the configuration parameter *Cache Allocation Size*. If you are using the L2 cache, you should set **System Cache** to Off. Check the setting of **Max MicroKernel Memory Usage**. When **Max MicroKernel Memory Usage** is set to a value greater than zero, you are using L2 cache.

If you are not using L2 cache, performance can be enhanced by turning on **System Cache**. The MicroKernel relies on the system cache to organize and group pages to be written. It delays a flush long enough to allow the system cache to write the pages to disk in a more efficient manner. However, if your server has an advanced self-cached disk array, you might achieve better performance by setting **System Cache** to Off.

For Windows Server only, you can use the Paging File and Process objects in the Windows Performance Monitor utility to determine whether the Windows system cache is being used effectively. For the NTDBSMGR instance, monitor the % Usage and % Usage Peak in the Page File object and the Page Faults/Second and Page File Bytes in the Process object.

**Performance Tuning**

Performance Tuning contains the following configuration settings:

- *Cache Allocation Size*
- *Communications Threads*
- *File Growth Factor*
- *Index Balancing*
- *Limit Segment Size to 2 GB*
- *Log Buffer Size*
- *Max Microkernel Memory Usage*
- *Number of Input/Output Threads*
- *Transaction Log Size*
Cache Allocation Size

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>1 megabyte to the amount limited by memory (see Note below)</td>
<td>Initialized dynamically at first start-up</td>
<td>megabytes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the size of the Level 1 cache that the MicroKernel allocates; the MicroKernel uses this cache when accessing any data files.

Speaking very generally, overall performance is usually best when the Cache Allocation Size is a value less than 40% of the physical memory on the system, and the Configuration setting Max MicroKernel Memory Usage is set to a value greater than 40%. Your exact optimal settings will depend on the size of your data files, the number of other applications running on the system, and the amount of memory installed in the computer.

**Server Engine**

On Windows, this setting is initially set to 20% of physical memory by the database engine the first time it starts and it writes that value to the Registry. After that, whenever the engine starts, it reads the value from the Registry. Changing the value using Configuration updates the value in the Registry. If you add or remove memory from the system, you must modify this setting to take best advantage of the new amount of memory available.

To optimize your performance, allocate a cache size no larger than the sum of the sizes of the files you are using. However, be careful not to take all available memory, especially when the server is running other applications. You cannot improve performance—and may waste memory—by specifying a value higher than you need.

**Workgroup Engine and Client Cache**

The database engine initially sets this value the first time it starts and writes the value to the Registry. The initial value is set to 20% of physical memory. The maximum size of the cache depends on the amount of memory in the system, but the engine sets an initial maximum of 64 MB. After the Registry setting is initialized, whenever the engine starts, it reads the value from the Registry. The engine never re-calculates the setting. Changing the value using Configuration updates the value in the Registry. If you add or remove memory from the system, you must modify this setting manually to take best advantage of the new amount of memory available.

**Client Cache**

This information also applies to the client software (Client cache) if the configuration setting Use Cache Engine is turned on.

**Note** If you use PSQL Clients prior to PSQL v10, the value for cache allocation size must be specified in bytes, with a minimum of 64 KB (65,536 bytes). The maximum is limited by the amount of memory.
Communications Threads

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>num_cores to 256</td>
<td>num_cores, where num_cores is the number of processors in the machine on which the database engine is running</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies how many threads the MicroKernel initially spawns to handle requests from remote clients. Communication threads are the elements that actually perform Btrieve operations on behalf of the requesting remote client process. In this way they are very similar to worker threads. Communications threads increase dynamically as needed up the maximum range allowed. The maximum is 256.

The Communications Threads setting can help improve scaling under certain conditions. For example, if you have many clients performing operations (typically writes) on one file, a lower setting should improve scalability. The lower number of threads prevents context switching on system resources. Another condition that this setting may improve is a slowdown caused by thrashing among large numbers of worker threads. Worker threads are dynamically created only if all the existing threads are waiting on record or file locks.

File Growth Factor

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0-100</td>
<td>15</td>
<td>Percent</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This value specifies the approximate percentage of free pages to maintain in version 8.x and later format data files. The setting does not apply to any previous file format versions. The MicroKernel uses this value to decide whether to extend a file or use free pages first. The database engine uses turbo write accelerator so the performance of disk writes can be expected to improve as the number of free pages increases within the file. This improvement is due to the engine's ability to write multiple contiguous file pages on disk. Thus, disk write performance is a trade-off against file size. However, maintaining too many free pages in a file can actually reduce overall performance.

To maintain a certain amount of contiguous free pages in a data file, the database engine must periodically expand the file. Keep in mind that the file size effects of this setting are exponential. For example, if you start with a file that has no free pages and you specify a File Growth Factor value of 50%, the file will eventually double in size. If you specify a File Growth Factor value of 75%, the file will quadruple in size. A value of 90% will cause the file to grow by as much as 10 times.

Note that only completely unused pages are counted as empty, so 15% empty pages does not mean that 15% of the file is unused, as some pages may not be completely full.

This value is only a hint to the MicroKernel. Depending on how heavily a given file is being updated, the actual percentage of empty space may be much less at any given moment.

This setting is not applicable to pre-8.x format files. These older files also have empty pages and the percentage of empty pages varies with the activity on the file.
Index Balancing

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting controls whether the MicroKernel performs index balancing. Index balancing increases performance on read operations; however, when you enable this option, the MicroKernel requires extra time and may require more disk I/O during insert, update, and delete operations. For more information about index balancing, refer to PSQL Programmer’s Guide available in the Developer Reference.

Limit Segment Size to 2 GB

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies that a data file is to be automatically broken into 2 GB operating system file segments as its size passes that boundary. If set to On, this setting specifies that you want data files divided into 2 GB segments. If set to Off, this setting specifies that data files will remain a single, nonsegmented file. The advantage of using a larger nonsegmented file is more efficient disk I/O. Therefore, you can expect increased performance.

Any nonsegmented files are subject to the limit on file size specified by your operating system. If a previously created file is already segmented, that segmentation remains on the file.

See also Automatic Upgrade of File Version for related information.

Log Buffer Size

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>262144 - limited by memory</td>
<td>The smaller value of the following: (physical memory in MB / 256) or 8 MB</td>
<td>bytes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the size of both the transaction log buffer and the archival log buffer that the MicroKernel uses. You can enhance performance by increasing the log buffer size, because the MicroKernel writes the log information to disk less frequently.

Note: If you set Log Buffer Size to a value greater than that of Transaction Log Size, then the MicroKernel automatically increments Transaction Log Size to the value you specified for Log Buffer Size.
**Max MicroKernel Memory Usage**

This setting specifies the maximum proportion of total physical memory that the MicroKernel is allowed to consume. L1, L2, and all miscellaneous memory usage by the MicroKernel are included (Relational Engine is not included). The database engine uses less if the specified proportion is not needed or not available.

If the value zero (0) is specified, then dynamic caching is turned off. In this case, the only cache available is L1, the size of which is specified by **Cache Allocation Size**. If you have a dedicated database server machine, then you should set **Max MicroKernel Memory Usage** to the maximum value, or whatever proportion of memory is not occupied by the operating system. If you run other applications on your database server and you need to balance performance between all of these, then you should set this value lower so that the database cache does not compete as much with the other applications when using available memory.

**Note** If **Cache Allocation Size** is set to a higher amount of physical memory than **Max MicroKernel Memory Usage**, then Cache Allocation Size takes precedence. For example, a machine has 1 GB of physical memory and you set Cache Allocation Size to 600 MB and Max MicroKernel Memory Usage to 40%. The L1 cache is allocated 600 MB of memory. Since Cache Allocation Size is higher, it takes precedence and the amount of memory allocated to the L1 cache exceeds the value specified for Max MicroKernel Memory Usage.

Use the following equation to determine the approximate value for **Max MicroKernel Memory Usage**.

\[
\text{Max MicroKernel Memory Usage} = \left( \frac{\text{L1_cache_size} + \text{internal_allocations} + \text{L2_cache_size}}{\text{size_of_physical_memory}} \right) \times 100
\]

where:

- **L1_cache_size** is the **Cache Allocation Size**
- **internal_allocations** is approximately 25% of the size of the L1 cache
- **L2_cache_size** is the amount of memory that expands and contracts based on memory load of the system
- **size_of_physical_memory** is the amount of memory installed in the machine

For more information on tuning performance, see **Tuning Performance**.

**Number of Input/Output Threads**

This setting specifies how many background I/O threads the MicroKernel spawns. These threads are responsible for writing all pages from the MicroKernel’s cache to the file on disk in an atomic and consistent manner. They also are responsible for initially opening a file and reading the File Control Record. Most of the other reads are done by local worker threads and communication threads. When the
MicroKernel updates or writes to data files, it assigns each file to a particular I/O thread sequentially. When it reaches the last thread, the MicroKernel starts over until all data files have been assigned to a background thread. Because the MicroKernel does not spawn additional I/O threads as needed, specify the maximum number of I/O threads you anticipate needing.

For best performance, set this value based on the average number of open files. Monitor shows the current and peak number of files open. If your database has an average of 256 files open, then the default of 32 I/O threads makes each thread responsible for 8 files. A good rule of thumb is to have about 8 files per I/O thread. For example, if your average number of open files is 400, you should use about 50 I/O threads. Specifying a value higher than 64 may degrade performance, but that depends on the capabilities of the system.

---

**Note** No accurate way is available to calculate the appropriate number of I/O threads because this setting depends on the machine's characteristics, OS configuration, and the database engine's planned work load.

---

### Transaction Log Size

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>65536 - limited by disk space</td>
<td>2 times the Log Buffer Size</td>
<td>bytes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

This setting specifies the maximum size of a transaction log segment. When the log file reaches its size limit, the MicroKernel closes the old log segment file and starts a new one. You might want to limit the size of your transaction log segments, because this reduces the disk space that the MicroKernel uses temporarily. However, limiting the size of the transaction log segments does require more processing by the MicroKernel and can degrade performance, because it has to close and create log segments more frequently.

---

**Note** If you set the value for this option less than the value you specified for Log Buffer Size, the Database Engine automatically adjusts Transaction Log Size by setting it to the value of Log Buffer Size.
Windows Client Configuration Parameters

The client configuration options are available in all the different installation setups. These options must be configured separately for each workstation, which includes servers acting as workstations.

You can configure the client using the graphical utility PSQL Control Center or the command-line interface utility `bcfg`. For PCC, see Using PSQL Control Center in PSQL User's Guide. For `bcfg`, see Configuration Through CLI Utility.

The following table is a mapping of all the available client configuration options and their settings.

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Gateway Durability</td>
</tr>
<tr>
<td></td>
<td>Number of Load Retries</td>
</tr>
<tr>
<td></td>
<td>Use IDS</td>
</tr>
<tr>
<td></td>
<td>Use Local MicroKernel Engine</td>
</tr>
<tr>
<td></td>
<td>Use Remote MicroKernel Engine</td>
</tr>
<tr>
<td></td>
<td>Wire Encryption</td>
</tr>
<tr>
<td></td>
<td>Wire Encryption Level</td>
</tr>
<tr>
<td>Cache Engine</td>
<td>Allocate Resources at Startup</td>
</tr>
<tr>
<td></td>
<td>Back to Minimal State if Inactive</td>
</tr>
<tr>
<td></td>
<td>Cache Allocation Size</td>
</tr>
<tr>
<td></td>
<td>Max MicroKernel Memory Usage</td>
</tr>
<tr>
<td></td>
<td>Minimal State Delay</td>
</tr>
<tr>
<td>Cache Engine Debugging</td>
<td>The settings available under Cache Engine Debugging perform the same functions for the client cache as similar settings under Server. See the related server settings for Debugging.</td>
</tr>
<tr>
<td>Communication Protocols</td>
<td>Enable Auto Reconnect</td>
</tr>
<tr>
<td></td>
<td>Supported Protocols</td>
</tr>
<tr>
<td></td>
<td>Connection Timeout</td>
</tr>
<tr>
<td>Performance Tuning</td>
<td>Use Cache Engine</td>
</tr>
<tr>
<td>Security</td>
<td>Runtime Server Support</td>
</tr>
<tr>
<td>Application Characteristics</td>
<td>Embedded Spaces</td>
</tr>
<tr>
<td></td>
<td>Verify Key Length</td>
</tr>
<tr>
<td></td>
<td>Splash Screen</td>
</tr>
</tbody>
</table>
Access
Access contains the following configuration settings:

- Gateway Durability
- Number of Load Retries
- Use IDS
- Use Local MicroKernel Engine
- Use Remote MicroKernel Engine

Gateway Durability

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This option specifies whether the MicroKernel Router should store in the registry a list of computers that do not have PSQL running on them. This decreases the time it takes to find a gateway engine. You must set this option to Off when new engines are added to the workgroup.

Number of Load Retries

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0 to 65536</td>
<td>5</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting specifies the number of times the MicroKernel Router attempts to connect to the target engine.

Use IDS

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting is primarily for use with legacy applications that used PSQL (IDS). IDS functionality is now integrated into the core product and IDS is no longer available as separately installed components. The integration of IDS requires that you reconfigure your client/server environment.

Typically, an application provides its own file location information. As an alternative, IDS provided file location mapping based on information in a text file, idshosts.

If your applications do not use the mapping feature through idshosts, set Use IDS to Off to improve performance.

If your applications already use idshosts, or if you prefer to use this alternative method to map file locations, set Use IDS to On. See Using the idshosts File.
Note PSQL 8.5 or later is required if you set **Use IDS** to **On** or if your legacy applications pass file location information in the format of a PIDS URL. The requester uses database URIs to represent the IDS information. Database URIs were added with PSQL 8.5. See *Database URIs* in PSQL Programmer's Guide available in the Developer Reference.

### Use Local MicroKernel Engine

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On, Off</td>
<td>On</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting determines whether a local application tries to connect to a local engine. If set to **Off**, no attempt is made to connect to a local engine.

### Use Remote MicroKernel Engine

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On, Off</td>
<td>On</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting specifies whether the MicroKernel Router allows access to a Server or Workgroup engine running on a remote server. If this value is set to **On**, and **Use Local MicroKernel Engine** is set to **On**, the remote server is tried first.

### Wire Encryption

See *Wire Encryption*.

**Note** Client-side wire encryption settings are not used by the PSQL JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See *Connection String Overview* in JDBC Driver Guide and *Defining Basic Connection Strings* in Data Provider for .NET Guide.

### Wire Encryption Level

See *Wire Encryption Level*.

**Note** Client-side wire encryption settings are not used by the PSQL JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See *Connection String Overview* in JDBC Driver Guide and *Defining Basic Connection Strings* in Data Provider for .NET Guide.
**Cache Engine**

These settings apply only when the cache engine is running. The Workgroup engine doubles as a cache engine. Note, however, that the cache engine is not used if a database server engine is running.

Cache Engine contains the following configuration settings:

- Allocate Resources at Startup
- Back to Minimal State if Inactive
- Cache Allocation Size
- Max MicroKernel Memory Usage
- Minimal State Delay

### Allocate Resources at Startup

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Database Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting instructs the cache engine to allocate resources, including threads and memory buffers, when the cache engine is started.

If you turn this option off, the cache engine does not allocate any resources until the first operation request. PSQL components automatically allocate resources as needed. Therefore, in most cases you do not need to do so explicitly.

### Back to Minimal State if Inactive

This setting displays only if the Workgroup engine is running.

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Database Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting causes the cache engine to free considerable memory and thread resources to the system and return to a minimal state after a certain amount of time without any active clients. The time interval is specified by the value of Minimal State Delay. The cache engine reallocates resources when another client becomes active.
Cache Allocation Size

This setting specifies the size of the Level 1 cache that the MicroKernel allocates; the MicroKernel uses this cache when accessing any data files.

Speaking very generally, overall performance is usually best when the Cache Allocation Size is a value less than 40% of the physical memory on the system, and the Configuration setting Max cache engine Memory Usage is set to a value greater than 40%. Your exact optimal settings will depend on the size of your data files, the number of other applications running on the system, and the amount of memory installed in the computer.

The database engine initially sets this value the first time it starts and writes the value to the Registry. The initial value is set to 20% of physical memory. After the Registry setting is initialized, whenever the engine starts up, it reads the value from the Registry. The engine never re-calculates the setting.

Changing the value using Configuration updates the value in the Registry. If you add or remove memory from the system, you must modify this setting manually to take best advantage of the new amount of memory available.

Note If you use PSQL Clients prior to PSQL v10, the value for cache allocation size must be specified in bytes, with a minimum of 64 KB (65,536 bytes). The maximum is limited by the amount of memory.

Max MicroKernel Memory Usage

This setting specifies the maximum proportion of total physical memory that the cache engine is allowed to consume. L1, L2, and all miscellaneous memory usage by the cache engine are included. The database engine uses less if the specified proportion is not needed or not available.

If the value zero (0) is specified, then dynamic caching is turned off. In this case, the only cache available is L1, the size of which is specified by Cache Allocation Size.

For more information on tuning performance, please see Tuning Performance.
Minimal State Delay

This setting displays only if the Workgroup engine is running.

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Database Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Numeric</td>
<td>0 - 2147483647</td>
<td>300</td>
<td>milliseconds</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting specifies how long the cache engine waits during a period of inactivity before returning to a minimal state. (This is the initial state in which the cache engine begins.) By returning to a minimal state, the cache engine frees considerable memory and thread resources to the system. In some cases, you may not want the cache engine to return to a minimal state. For example, you may be running a batch file that uses the cache engine repeatedly. The cache engine reallocates resources when another client becomes active.

This setting is ignored if the value of Back to Minimal State if Inactive is set to Off (the default).

Cache Engine Debugging

These settings apply only when the cache engine is running. The Workgroup Engine doubles as a cache engine. Note, however, that the cache engine is not used if a database Engine is running.

The settings available under Cache Engine Debugging perform the same functions for the Client cache as similar settings under Server Debugging perform for the main database engine. For more information about each setting, see the related server setting:

- Number of Bytes from Data Buffer
- Number of Bytes from Key Buffer
- Select Operations
- Trace File Location
- Trace Operation

Communication Protocols

Communication Protocols contains the following configuration settings:

- Enable Auto Reconnect
- Supported Protocols
- Connection Timeout

Enable Auto Reconnect

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting specifies whether you want the client to attempt to auto-reconnect during a network outage. A setting of On means Auto Reconnect is enabled.

Auto Reconnect is not in effect unless this setting is also enabled in the server configuration.
Supported Protocols

This setting specifies the protocols that are used by the client. If more than one protocol is specified, the client attempts to connect using all available protocols. When the first protocol succeeds, that protocol is used for the remainder of the session. The available options are:

- TCP/IP
- SPXII
- NetBIOS

**Note** You must have at least one protocol enabled at both the client and the server or they cannot communicate. NetBIOS is valid only for PSQL Workgroup, not PSQL Server. TCP/IP is the only supported protocol for PSQL running on a Linux platform. Therefore, the Supported Protocols setting is not available on Linux.

Connection Timeout

This setting specifies how long the client waits while searching for or connecting to a remote database engine. If this value is set too low, the client may return spurious “server not found” errors, because it is timing out before it has a chance to complete the connection. If the value is set too high, when the client attempts to connect to a server that is not reachable, you may encounter lengthy delays before receiving an error. Generally speaking, a value between 15 and 30 seconds is adequate for most networks. If you receive many “server not found” errors, try a higher setting.

This setting was previously named: TCP/IP Timeout for Communication Requester.

**Performance Tuning**

Performance Tuning contains the following configuration setting:

- Use Cache Engine
Use Cache Engine

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Database Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting specifies whether the Client Cache Engine should be used. The Client Cache Engine is a specialized version of the MicroKernel Engine that caches data for reading purposes and runs as a separate process. For simplicity, Client Cache Engine is often referred to as the “Client cache.”

If **Use Cache Engine** is "off," nothing is cached on the Client side. READ requests from an application retrieve data from the remote database Engine.

If **Use Cache Engine** is "on," the Client Cache Engine acts as an intermediary between the Client and the remote database Engine to cache data. The first time an application issues a READ request, the Client Cache Engine caches the data. If another READ is issued again for the same record, the request can be satisfied by the Client Cache Engine though the cached data. The READ does not have to access the remote database Engine.

The Client cache is similar in many ways to the Workgroup Engine. By default, it auto-loads into memory when an application first accesses the database, and it unloads a few minutes after that application is stopped.

You may wish to keep the Client cache in memory at all times to avoid the performance cost of re-populating the cache with each usage session. If you want to keep the Client cache loaded, run the following command from a command prompt:

```
file_path\PSQL\bin\w3dbsmgr -btrv.
```

A tray icon appears after the Client cache starts. The tray icon allows you to control the Client Cache Engine from the Windows task bar. To stop the Client cache, right-click the tray icon then click **Stop Engines and Exit**.

If the Client is installed as a service, the Client Cache Engine service is set by default to auto-start. However, even though the Client cache service is running, an application does not use the Client cache unless **Use Cache Engine** is set to "on."

---

**Note** PSQL synchronizes the Client cache with the database Engine cache and other Client cache locations. This behavior is fully automatic and entirely transparent. However, there is a maximum delay of 5 seconds between any database change happening in the database Engine cache and it being reflected in all Client cache locations. If the possibility of such stale pages existing in the cache for a maximum of 5 seconds is unacceptable in your system, do not use the Client cache.

The following operations are **not** stored in the Client cache:

- everything inside a transaction
- operations with a lock bias
- write operations such as INSERT, UPDATE, and DELETE
- open and close operations

See also the discussion of Client cache under the setting **Cache Allocation Size**.
Security

Security contains the following configuration setting:

- **Runtime Server Support**

**Runtime Server Support**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>String</td>
<td>Yes</td>
<td>Yes</td>
<td>None</td>
<td>Not applicable</td>
</tr>
<tr>
<td></td>
<td>No</td>
<td></td>
<td>user_name,password</td>
<td></td>
</tr>
</tbody>
</table>

This setting controls runtime server support. If enabled with the value **Yes**, the current user name and password for the drive on which you are presently running are used. To enable RTSS with a different username, enter “user_name,password.”

Note that you may use a fully qualified NDS user name in the format CN=user_name.O=organization,password. The user name may also be a simple Bindery name. The first entry in the Bindery context list must contain the simple name or the NDS login fails. If the NDS login fails for a simple name, a Bindery login is attempted. The Bindery login may cause delays while the login attempt is processing.

SUPERVISOR and ADMIN are not valid user names, even if supplied with the correct password. If the requester cannot find a login user name other than SUPERVISOR or ADMIN, it does not attempt to log in.

**Application Characteristics**

Application Characteristics contains the following configuration settings:

- **Embedded Spaces**
- **Splash Screen**
- **Verify Key Length**

**Embedded Spaces**

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On Off</td>
<td>On</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This option instructs the MicroKernel Engine to allow embedded spaces in file names for MicroKernel Engine operations.
Splash Screen

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting controls whether or not the MicroKernel Engine splash screen displays. The splash screen displays the first time a client requester loads.

Verify Key Length

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This option can be used for legacy Btrieve applications to prevent the requester from verifying that the index key length passed to the client requester is large enough to hold the key. Setting this option to “off” may allow such applications to avoid status 21 errors.

Caution If set to “off,” this option disables the check by the PSQL requester to prevent memory overwrites. A memory overwrite can cause a general protection fault (GPF) among other undesirable conditions.
Linux Client Configuration Parameters

You can configure both PSQL Clients and Servers using the graphical utility PSQL Control Center or the command-line interface utility `bcfg`. For PCC, see Using PSQL Control Center in PSQL User’s Guide. For `bcfg`, see Configuration Through CLI Utility.

Case of Configuration Values

When checking or editing the values of settings, the Linux client performs a case-insensitive comparison. For example, entering ‘Yes’ or ‘yes’ for a setting value is interpreted identically by the Linux client.

Client Performance Affected by “Local” Setting

When the Linux client interface is first invoked, it populates its default settings in the PSQL registry. The PSQL Client does not have knowledge on whether its installation includes a server engine or not. Therefore, it sets the “Local” setting to yes. This can have an impact on the performance of your Linux client.

If the machine on which you are using the client does not have a server engine, you should set the Local setting to no. See Use Local MicroKernel Engine.

File Names with Embedded Spaces

By default, the Linux client interface does support file names that contain embedded spaces.

For example:
```
/mymount/usr/gary/file with spaces.mkd
```

If you want to use file names without embedded spaces, you need to change the “Embedded Spaces” setting. See Embedded Spaces.

Configuration Reference

The following table lists the configuration options for the Linux client.

<table>
<thead>
<tr>
<th>Configuration Option</th>
<th>Parameter Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Access</td>
<td>Use Local MicroKernel Engine</td>
</tr>
<tr>
<td></td>
<td>Use Remote MicroKernel Engine</td>
</tr>
<tr>
<td></td>
<td>Use IDS</td>
</tr>
<tr>
<td></td>
<td>Wire Encryption</td>
</tr>
<tr>
<td></td>
<td>Wire Encryption Level</td>
</tr>
<tr>
<td>Communication Protocols</td>
<td>Enable AutoReconnect</td>
</tr>
<tr>
<td>Application Characteristics</td>
<td>Embedded Spaces</td>
</tr>
<tr>
<td></td>
<td>Verify Key Length</td>
</tr>
</tbody>
</table>
Access

Access contains the following configuration settings:

- Use Local MicroKernel Engine
- Use Remote MicroKernel Engine
- Use IDS
- Wire Encryption
- Wire Encryption Level

Use Local MicroKernel Engine

See Use Local MicroKernel Engine.

Use Remote MicroKernel Engine

See Use Remote MicroKernel Engine.

Remote Engine and UNC Paths

For UNC paths to work properly from a client, the following steps must be performed:

- You must be running an engine on the same computer as the file that you are trying to access.
- You must set Use Remote MicroKernel Engine to “on.”

\[\textbf{Note} \] You cannot send a UNC path that points to the local Linux machine. However, you can use a path that is in the UNC style such as \\
//localhost/usr/local/psql/data/samples/sample.btr

If you do not want an engine on your file server (that is, you want to use the client's local engine), then you will need to mount the remote file system on the client, and modify the path so that it is a "native format" path and not UNC format. For example, the following path is a native Linux format:

/mnt/myremotedata/sample.btr

Use IDS

See Use IDS.

Wire Encryption

See Wire Encryption.

\[\textbf{Note} \] Client-side wire encryption settings are not used by the PSQL JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See Connection String Overview in JDBC Driver Guide and Defining Basic Connection Strings in Data Provider for .NET Guide.
Wire Encryption Level
See Wire Encryption Level.

Note: Client-side wire encryption settings are not used by the PSQL JDBC and ADO.NET access methods. For them, encryption can be specified using the connection string. See Connection String Overview in JDBC Driver Guide and Defining Basic Connection Strings in Data Provider for .NET Guide.

Communication Protocols
Communication protocols contains the following configuration setting:
- **Enable AutoReconnect**

Enable AutoReconnect

<table>
<thead>
<tr>
<th>Type</th>
<th>Range</th>
<th>Default</th>
<th>Units</th>
<th>Requires Engine Restart</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boolean</td>
<td>On</td>
<td>Off</td>
<td>None</td>
<td>Not applicable</td>
</tr>
</tbody>
</table>

This setting specifies whether you want the client to attempt to auto-reconnect during a network outage. A setting of on means Auto Reconnect is enabled.

Auto Reconnect is not in effect unless this setting is also enabled in the server configuration.

Note: The PSQL Linux client supports this auto-reconnect feature, but currently the Linux server does not. Therefore, you can only use the AutoReconnect (PARC) feature from a Linux client connecting to Windows servers.

Application Characteristics
Application characteristics contains the following configuration settings:
- **Embedded Spaces**
- **Verify Key Length**

Embedded Spaces
See Embedded Spaces.

Verify Key Length
See Verify Key Length.
Performance

Analyzing and Tuning Database Performance

This chapter covers the following topics:

- Analyzing Performance
- Tuning Performance
- Performance on Hypervisor Products
Analyzing Performance

PSQL Server for Windows provides performance counters for use with the Windows Performance Monitor. The performance counters measure state or activity of the database engine, which allows you to analyze performance of your application.

See Monitoring Performance Counters.
Tuning Performance

This section provides some general tips on how to maximize performance on the initial database connection and on runtime operations. While we can offer some general guidelines, the performance of any specific application depends on a great number of factors, including but not limited to the following:

- Network bandwidth and use
- Coding techniques in the application
- Physical storage space available
- Memory available
- Processor speed
- Application usage patterns (such as write-heavy, read-heavy, transactions used/not used, small record size, large record size, complex queries, simple queries, ODBC, Btrieve only, and so forth)
- Unrelated applications competing for CPU cycles
- Database engine configuration

As you can see, the engine configuration plays a relatively limited role in the overall performance of any given application. Further, the database engine dynamically manages a variety of resources based on usage patterns. It tunes itself to your environment as needed. The following sections provided are offered only as helpful guidelines and are not a guarantee of any specific level of performance.

Spotting Performance Bottlenecks

You can use Monitor utility to expose performance bottlenecks related to certain database engine configuration options. You can start Monitor from the operating system Start menu or Apps screen or from the Tools menu in PSQL Control Center.

Monitor Displays and Configuration Parameters

Two different Monitor menu selections display performance readings related to configuration options:

- MicroKernel > Resource Usage
- MicroKernel > Communications

The database engine dynamically manages several server configuration options, as shown in the following table.

<table>
<thead>
<tr>
<th>Dynamically Managed Setting</th>
<th>Value Displayed in Resource Usage Window</th>
<th>Value Displayed in Communications Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Handles</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Clients</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Worker Threads</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Total Remote Sessions</td>
<td>✓</td>
<td></td>
</tr>
</tbody>
</table>
Interpreting the Displays and Taking Action

You can make use of the information displayed in Monitor. Monitor displays three pieces of information about each type of resource. For example, the Total Remote Sessions display shows:

- **Current.** The current number of actual remote sessions operating.
- **Peak.** The highest number of actual remote sessions that have been used since the engine was started.
- **Maximum.** The maximum number of remote sessions allowed.

If the Peak value for a resource is the same as the Maximum value, then you may want to set the configuration property to increase the Maximum value for the resource, thus allowing the database engine to allocate additional instances of that particular resource when it needs to.

**Before You Modify Configuration Parameters**

The following sections assume the following:

1. **PSQL Control Center (PCC) is already open.**
   - If you need assistance with this task, see Starting PCC on Windows of PSQL User's Guide.
2. **You have already registered (if applicable) the engine you wish to configure.**
   - If you need assistance with this task, see To register a remote server engine of PSQL User's Guide.
3. **You have appropriate operating system privileges to configure the given engine.**
   - If you need assistance with this task, see Granting Administrative Rights for the Database Engine of PSQL User's Guide.
4. **For some engine settings, you must restart the database engines after making configuration changes.**

**Minimizing Initial Connection Time**

The theory underlying minimal connection time revolves around three requirements. These requirements are summarized next, and detailed procedures follow:

- **Known communications protocol.** You do not want the client or server to spend additional time trying to connect via protocols that are not supported in your environment. By removing client/server support for unavailable protocols, you prevent the networking components from trying to use them.

- **Known location of database engine.** You do not want the client to attempt to connect to an engine that does not exist. By specifying Workgroup-only or Server-only support, as appropriate, you prevent the client from timing out while attempting non-existent connections. In environments where you have both unshared and shared databases, you can use Gateway Durability to force the database engine to keep track of machines where no database engine is running, so that it never attempts to connect to an engine on these machines, but uses another method to access the data.

- **Database engine ready to execute.** When a sleeping engine receives a new connection request, it takes time to re-allocate resources and return to a runtime state. If connection speed is more important than resource usage on the server, you can prevent the server engine from sleeping when it is not in use.

**Client Parameters**

You must be at the client machine to change the client parameters. You must change the parameters at each workstation whose settings you wish to change.
➢ To minimize client-side connection delays

1 In PSQL Explorer, expand the Local Client node in the tree (click the expand icon to the left of the node).

2 Right-click MicroKernel Router.

3 Click Properties.

4 Click Communication Protocols in the tree.

5 For Supported Protocols, ensure that the desired protocols are selected (check marked) and the protocols not being used are not check marked.

6 Click Apply.

The client is now prevented from attempting to communicate on protocols that are not being used.

7 Click Access in the Properties tree.

8 If you are using only a remote Server or remote Workgroup engine, ensure that Use Local MicroKernel Engine is not check marked. (That is, it is set to Off.)

If you are using only a local Workgroup engine, ensure that Use Remote MicroKernel Engine is not check marked. (That is, it is set to Off.)

If you sometimes use a Workgroup engine and you sometimes connect to a Server engine or a remote Workgroup engine, you must leave both settings On (check marked).

In such a mixed environment with shared and unshared data, you can set Gateway Durability to On at each client. This setting forces the client software to keep a list of the names of any machines on which it is unable to connect to a database engine. In order for the client software to determine no engine exists on a given computer, it waits for all of its network protocol requests to time out.

If your data is stored on a server that does not have a PSQL database engine on it, and you have set Use Remote MicroKernel Engine to Yes, the client must time out at least once to discover that there is no engine on that machine. Gateway Durability ensures that this time-out only happens the first time your application tries to access that data.

Note Using Gateway Durability fixes the network topology. If you later install a Server or Workgroup engine on the remote computer, you must turn off Gateway Durability on each client so that the list of computers without database engines is deleted (thus allowing you to connect to the new database engine). You may turn Gateway Durability back on immediately, but it starts with an empty list.

9 Click OK.

The client is now prevented from attempting to connect to any database engine types that are not in use.

Server Parameters

➢ To minimize server-side connection delays

1 In PSQL Explorer, expand the Engines node in the tree (click the expand icon to the left of the node).

2 Right-click the database engine for which you want to specify configuration settings.
3 Click Properties.

4 For Supported Protocols, ensure that the desired protocols are selected (check marked) and the protocols not being used are not check marked.

5 Click Apply.

The server is now prevented from attempting to communicate on protocols that are not being used.

---

**Note** Ensure that at least one protocol you have selected for the Server configuration is the same one as selected for the Client configuration. Your client and server cannot communicate if they are not using the same protocol.

6 Click Memory Usage in the Properties tree.

7 Select Allocate Resources at Startup to set the value to On (check box selected).

   This option specifies that the database engine should allocate all necessary memory when it starts up, rather than when the first connection request comes in. Choosing this value requires more memory, but from the client perspective allows the engine to become operational faster.

8 Ensure that Back to Minimal State if Inactive is set to Off (check box cleared).

   This option specifies that the database engine should not release resources back to the operating system if the engine is inactive. All resources remain allocated and ready for use at the next client connection.

9 Click OK.

10 Click Yes to restart the engine for these changes to take effect.

---

**Maximizing Runtime Throughput**

The theory behind maximum throughput relies on too many variables to list here. Several of the most significant factors are:

- **Adequate physical memory.** If your host system has too little RAM, the system spends most of its time and CPU cycles swapping memory to disk and back, as different users and processes compete for limited resources.
- **Adequate CPU capacity.** If your CPU cannot keep up with the inflow of data processing requests, application performance suffers.
- **Adequate network bandwidth.** If your network is slow or suffers from a high collision rate, the database engine may sit idle between data access requests, while each client seems to display poor performance.
- **Minimal disk I/O.** Disk I/O is significantly slower than memory I/O. You want to avoid accessing the disk as much as possible, by having sufficient cache.
- **Adequate resource allocations.** Even with massive physical memory available, database performance may suffer if database engine configuration parameters are set artificially low.

In the end, optimal performance is a balancing act among network bottlenecks, disk I/O bottlenecks, memory bottlenecks, and CPU bottlenecks. This section provides some guidelines on how to reduce memory and disk I/O bottlenecks.
Fast Disk versus Fast CPU

If you want to maximize the effect of your hardware investment for performance gains, you must understand the existing constraints on your performance. If you have a database that is so large that you cannot reasonably buy and install enough memory to cache a significant part of the database, then performance is likely to be constrained by the disk I/O. Under these conditions, you may be better off to invest in a fast RAID disk array to maximize performance of the disk I/O.

In addition, if your application uses the Relational Engine and forces temporary files to be created frequently, you may want to ensure that the directory where these files are created is located on a fast disk drive. For more information about the location of this directory and the types of queries that generate temporary files, see Temporary Files in SQL Engine Reference.

If your database is small enough to be fully or near-fully cached in memory, then adding a fast disk array is unlikely to provide a significant performance boost. Under these conditions, upgrading the CPU or adding an additional CPU may provide the best performance improvement value.

Ensuring Adequate Physical Memory and Database Cache

Starting with Pervasive.SQL V8, the database engine offers Level 2 dynamic cache in addition to the Level 1 cache specified by the configuration setting, Cache Allocation Size. Assuming you do not turn off the Level 2 dynamic cache by setting Max MicroKernel Memory Usage to zero, the need to manually adjust the Level 1 cache size is much less critical than in previous releases. With that in mind, this section explains how to ensure that you have enough memory available for optimal performance of the database engine.

Ideally, your database engine should be able to allocate enough memory to cache full copies of every database it hosts, thus avoiding as much disk I/O as possible. Obviously, caching one or more entire databases is not practical in some situations, particularly when database size is very large. In addition, such measures as adding RAM to the machine only improve performance if the existing system resources are heavily loaded under normal usage.

The database engine dynamically selects a Level 1 cache size value when it starts up the first time. However, this value is based on available memory and may not be the ideal amount of cache for your environment.

➢ To calculate the ideal size of the database memory cache

1 Start by adding up the file sizes of all the data files serviced by the database engine.

Note If you have more than one database serviced by the engine, but they are never used at the same time, add up the file sizes of just the largest database.

For example, assume there are two databases on your server, with the following file sizes, and users access both databases at the same time:

<table>
<thead>
<tr>
<th>Database A</th>
<th>Database B</th>
</tr>
</thead>
<tbody>
<tr>
<td>file1.mkd</td>
<td>Afile.mkd</td>
</tr>
<tr>
<td>223 MB</td>
<td>675 MB</td>
</tr>
<tr>
<td>file2.mkd</td>
<td>Bfile.mkd</td>
</tr>
<tr>
<td>54 MB</td>
<td>54 MB</td>
</tr>
</tbody>
</table>
Performance

The sum of all these files is 1,430 MB.

The number you have now is the maximum amount of memory that the database engine would use if it cached all its hosted data. This number can be referred to as MaxCache.

You would never want to specify a value greater than this for Cache Allocation Size, because you would be allocating memory to the database engine that it would likely never use. A reasonable rule of thumb is to set Cache Allocation Size to about 20% to 70% of MaxCache. Lower values in this range are best for read-intensive applications, and higher values are best for write-intensive applications since all write/update operations take place in the Level 1 cache.

Note File pages are only written to the database cache when they are accessed. Thus, for a database engine to use MaxCache amount of memory requires every page in the database to be accessed. This system of estimating assumes a long-term steady state for database usage. If you bring the database engine down nightly or weekly, it may be unlikely that the database engine would access every page in the database within the given period of uptime.

If this situation applies to you, you may wish to estimate the average number of distinct pages that your application accesses within the given uptime period, multiply that by the page size, and obtain a more realistic value of MaxCache for your particular uptime scenario.

See also Counters for MicroKernel Cache.

On Windows 32-bit operating systems, all user processes are limited to 2 GB of memory. If you have calculated a value of MaxCache larger than 2 GB, and your database engine runs on a 32-bit Windows operating system, then you should make sure that the MaxCache value you select never allows the engine to exceed the 2GB boundary, or your engine may fail to allocate memory and subsequently fail.

To determine how much total physical memory you need

Use the following equation to determine the approximate amount of total physical memory required by the database engine.

Maximum Database Memory Usage = L1_cache_size + internal_allocations + L2_cache_size

where:

L1_cache_size is the Cache Allocation Size
internal_allocations is approximately 25% of the size of the L1 cache
L2_cache_size is the amount of memory that expands and contracts based on memory load of the system

Note the following:

<table>
<thead>
<tr>
<th>Database A</th>
<th>Database B</th>
</tr>
</thead>
<tbody>
<tr>
<td>file3.mkd</td>
<td>92 MB</td>
</tr>
<tr>
<td>file4.mkd</td>
<td>14 MB</td>
</tr>
<tr>
<td></td>
<td>Cfile.mkd</td>
</tr>
</tbody>
</table>
- The L1 cache is a fixed size based on Cache Allocation Size. It does not expand or contract based on database operations. If your application performs numerous WRITE operations, increase the L1 cache as much as possible.
- The greatest performance is obtained if all of the data can fit into the L1 cache. If all of the data will not fit into the L1 cache, adjust Cache Allocation Size and Max MicroKernel Memory Usage to use a reasonable amount of system memory.
- The L2 cache expands and contracts based on memory load of the system. For example, if other applications require more memory, the L2 cache contracts. If ample memory is available, the L2 cache reaches a maximum based on the equation above. The expansion and contraction affects performance. Contraction causes data pages to be removed from the L2 cache, for example. Reading the pages back from disk storage takes longer than if the pages could have been retained in the cache.
- The L2 cache contains compressed data pages (more pages in less space). Accessing pages from the L2 cache takes longer than from the L1 cache, but is faster than accessing the pages from disk storage. The L2 cache is helpful for applications that perform numerous READ operations but cannot fit all of the read data pages into the L1 cache.

**Minimizing Disk I/O**

Reading and writing data to/from disk is much slower than reading and writing to/from memory. Thus, one way to optimize performance is to minimize disk activity.

An important consideration in your attempts to minimize disk I/O is recoverability of data. Disk I/O is a direct trade off against transaction durability and recoverability. The more data you keep in memory without pausing to log changes to disk, the faster the database performs. On the other hand, the more data you keep in memory without pausing to log changes to disk, the more data you lose if the system experiences a failure.

➢ To reduce disk I/O

1. As discussed in the previous sub-section, Ensuring Adequate Physical Memory and Database Cache, one of the most important considerations is to ensure you have enough database memory cache to avoid frequently swapping data pages between disk and cache. See that section for details.

   One of the best ways to reduce disk I/O is to make sure that the dynamic Level 2 cache is turned on. The Level 2 cache adjusts its size dynamically as application usage changes, storing as much data as possible in memory and thus avoiding disk I/O when cache demands exceed the capacity of the Level 1 fixed cache. By default, the Level 2 cache is turned on. To verify that your database engine is using Level 2 cache, check the properties configuration setting Max MicroKernel Memory Usage (see Max MicroKernel Memory Usage).

2. Next, consider how much logging you require and what quantity of database operations you are willing to lose in a system failure. The greater the quantity of changes you are willing to lose, the more you can risk in the pursuit of performance.

   Using Archival Logging, Transaction Durability, and Transaction Logging all require log files. By default, archival logging is turned off. Turn it on only if you perform regular backups and you need the capability to restore data up to the moment of a system failure. When you specify the files to be logged, be sure to specify only the files for which you absolutely must have logging. See Chapter 9, Logging, Backup, and Restore, for more information.
By default, transaction logging is turned on. Turning off transaction logging should improve performance slightly, but does not guarantee multi-file consistency and transaction atomicity. Before turning off transaction logging, check with your application vendor to be sure they allow the application to run without this feature.

**Caution** The consistency of any multi-file database cannot be guaranteed if transaction logging is disabled.

By default, transaction durability is turned off. Turn on this feature only if your application requires completed transaction operations to be durable through a system crash. Transaction durability entails the highest performance penalty, and the trade off is the highest safety of your completed transactions.

3 If you have any logging features turned on, you can specify how much data the engine stores in memory before writing to disk. This feature is important because the changed data builds up over time. The more log data you allow to build up in memory, the less frequent the disk writes are.

The setting **Log Buffer Size** specifies the number of bytes of database operations that the engine stores in memory before writing them out to the log files. (Click **Performance Tuning** on the server Properties tree.)

If a system failure occurs, the data in the log buffer is lost.

4 If you have transaction durability turned on, you can specify the maximum size of the log segments on disk. Specifying a larger log segment size can improve performance slightly, because fewer log segments have to be created and closed over time.

The setting **Transaction Log Size** specifies the maximum number of bytes that can be stored in a log segment before closing it and opening a new segment. (Click **Performance Tuning** on the server Properties tree.)

5 If your database is highly used, consider configuring your system to maintain the logs on a separate physical volume from the volume where the data files are located. Under heavy load, performance is typically better when the writes to the log files and to the data file are split across different drives instead of competing for I/O bandwidth on a single drive. The overall disk I/O is not reduced, but the load is better distributed among the disk controllers.

6 If your application usage is weighted heavily in favor of database read operations, you can increase performance by turning on **Index Balancing**. (Click **Performance Tuning** on the server Properties tree.) Over time, index balancing increases the number of nodes on the average index page, allowing read operations to occur faster. However, for insert, update, and delete operations, additional time and disk I/O may be required because the engine balances the index nodes across adjacent pages.

7 Be sure that tracing is turned off, both in the MicroKernel and/or at the ODBC level. Tracing may cause a significant reduction in performance because it can introduce a large amount of disk I/O.

To ensure ODBC tracing is turned off, start **ODBC Administrator** from PSQL Control Center. In ODBC Administrator, click the **Tracing** tab. If tracing is off, you should see a button labeled “Start Tracing Now,” and thus you should click **Cancel**. If tracing is on, click **Stop Tracing Now**, then click **OK**.

To ensure MicroKernel tracing is turned off, set the properties configuration **Trace Operation** to **Off** (not check marked). (Click **Debugging** on the server Properties tree.)
Ensuring Adequate Resource Allocation

If your database server platform has adequate memory and CPU power, you should ensure that your database engine can take full advantage of the available hardware resources to service multiple clients and multiple data files most efficiently.

➢ To configure multiple client and file handling

1. The setting **Number of Input/Output Threads** allows you to specify how many threads are available to handle file operations. (Click **Performance Tuning** on the server Properties tree.)

   As a guideline, the value of this setting should be about 1/8 the number of files the application has open, on average. For example, if the application has 40 files open most of the time, I/O Threads should be set to 5.

   Using Monitor, click **MicroKernel > Resource Usage** from the menu. In the window that appears, the **Files** display shows you current and peak number of files open. You can generate an average reading by recording several Current values over time. Then you can specify an appropriate setting for I/O Threads based on the average value.

**Large System Cache**

Some Windows operating systems provide a Large System Cache setting that allows the system to take advantage of free memory to cache recently accessed data. By default on certain server editions, this setting is “on,” which favors simple file sharing and can result in very aggressive system cache growth.

The aggressive use of memory for file caching can swap out PSQL, which can cause a substantial performance issue with the database. You may notice a performance decrease if Large System Cache is “on,” and the PSQL setting **System Cache** is also set to “on” (either explicitly or you have set **Max MicroKernel Memory Usage** to zero). One possible solution to increase database performance is to turn off Large System Cache.

To turn off the cache, access the system properties (from example, from the Control Panel or from the properties for My Computer). Click the **Advanced** tab, then the **Settings** button for “Performance.” On “Performance Options,” click the **Advanced** tab.

Under “Memory Usage,” if the **System Cache** option is selected, Large System Cache is turned “on.” To turn it “off,” click the other option for “Memory Usage:** Programs. Click **OK** as required to close the series of open dialogs, then restart the operating system. Note that a reboot is required for the operating system setting to take effect.
Performance on Hypervisor Products

To achieve the best performance for PSQL, ensure the following:

- Adherence to the performance best practices recommendations from the hypervisor vendor.
- The VM hosting PSQL has ample memory (RAM).
- The hypervisor host has enough virtual CPUs to minimize virtual CPU contention among all of the VMs on the same machine. This prevents contention with the VM running PSQL. If you use affinity rules, ensure that all cores are running on the same socket.
- The PSQL data files reside on very fast physical storage and minimize spindle contention and controller contention for the physical storage device.
Database Globalization

Globalization Features of PSQL

This chapter contains the following sections:

- Overview
- Concepts and Definitions
- Choosing a Character Set and Encoding
- Multilingual Database Support With Unicode UTF-8
- Multilingual Database Support With Unicode UCS-2
- Multilingual Database Support with Legacy and OEM Encodings
- Database Code Page and Client Encoding
- Unicode Support in PSQL Utilities
- Support for Collation and Sorting
- Locale Support
Overview

Globalization means, in this context, adapting computer software to different languages. It is now commonplace that data be accessed by users around the globe and that applications present the data in the user’s own language. Support for globalization in PSQL allows your application to store text in multiple languages in the same database. This means that your application can store, process, and retrieve data in whatever language is required.

This chapter explains the PSQL features with which you can support globalization for your applications. It discusses overall approaches to globalization and the particular PSQL features that support globalized applications. By default, PSQL remains backward compatible with legacy text encodings and this chapter also discusses the settings that ease globalization for applications using legacy encodings.
Concepts and Definitions

This section presents important concepts and definitions of terms used in this chapter.

Character Sets

A **character set** defines the list of text and other symbolic characters that are recognized by a given hardware and software system. For a system that only needs to recognize the characters used in English, the set can be as small as the letters A-Z and a-z, the numerals 0-9 and a few punctuation symbols. Support for additional languages increases the size of the character set. For example, European languages add characters with accents and other diacriticals. Other languages have completely different characters.

Legacy Character Sets

A character is represented in a computer system as a numerical value (getting from a character to a number and back is called encoding and is discussed below). To represent all of the characters in a large character set requires a large range of numerical values. In order to efficiently use what was once limited and expensive storage resources, computer systems have commonly used a byte (8-bits) to store character representations. This limited the size of the character set to 256 characters. Consequently, we now have a legacy of specialized character sets for representing specific languages, such as Japanese, or limited groups of languages, such as Western European. PSQL supports a number of these legacy character sets.

The Unicode Character Set

The Unicode standard defines a character set that contains every character used in spoken languages in the world (see [www.unicode.org](http://www.unicode.org)). Unicode also expands the concept of a character set by defining additional annotation information to specify letter spacing, right-to-left behavior, word and line breaks, and so forth. This allows applications to properly display and manipulate Unicode text. Applications, and the database, also need this additional information for such actions as case conversion and sorting. PSQL recognizes the Unicode character set, providing support for character data storage and retrieval in whatever languages are required by the application.

Encoding

Encoding is the association of each character in a character set with a numerical value. Initially, computer systems and system programming languages did not distinguish between characters and bytes and encoding was simply the byte value corresponding to a particular character. In order to respond to the need to display more characters than would be possible to encode in a single byte, different encodings have been defined. Encodings for larger character sets may use multiple bytes per character.

Legacy Encodings

For legacy character sets, the encoding is defined in a **code page**. You can think of the code page as the lookup table for converting from a character to a value (or a value to a character). It is important that applications that use text always use the same code page. A character that is stored in the database using one code page may be displayed as a different character when read using a different encoding.

The binary unit of all legacy code pages is an 8-bit byte. Most legacy code pages currently in use are supersets of ASCII (American Standard Code for Information Interchange), a 7-bit code defining a
character set consisting of 128 control codes and printable characters. By using the 8th bit an additional 128 codes are possible for a total of 256 character encodings in a single byte value.

Microsoft Windows has two groups of code pages, ANSI and OEM code pages. Although code pages in both of these groups are extended ASCII code pages, neither group distinguishes between characters and bytes.

In ANSI (American National Standards Institute) code pages, the non-ASCII values (values greater than 127) represent international characters that are usually customized for a language or group of languages. ANSI code pages are typically used for byte string applications that use a graphical user interface on Windows systems. ANSI code pages are also sometimes referred to as active code pages (ACP, referred to as the client code page in PSQL). A Windows application always has one currently active ANSI code page. For example, the default active code page for English on Windows is code page 1252.

OEM (original equipment manufacturer) code pages are, as the name implies, code pages developed by a given manufacturer for specific systems. These code pages were originally used for MS-DOS and are still used for console applications. The usual OEM code page for English is code page 437. A typical OEM code page has a character set similar to one of the ANSI code pages, but uses a different character order for values greater than 127.

Unicode Encodings

In the Unicode character set, each character is assigned a unique value called a code point. That code point value is then encoded for storage. The code points are organized into planes. Each plane can contain 65536 code points. The first plane, plane 0, is named the Basic Multilingual Plane (BMP) and contains the majority of the code points currently defined. (Unicode has provision for up to 17 planes. At the time of this writing, only the first six contain code points.) The Unicode standard has several methods of encoding the code points. Two that are commonly used are UTF-8 and UCS-2. UTF-8 encodes character code point values to a bytestring using one to four bytes per character. UCS-2 encodes character code point values using 16-bit values, often referred to as wide characters.

PSQL recognizes the Basic Multilingual Plane code points and is compatible with applications that use the Unicode encodings UTF-8 for byte strings and UCS-2 for wide character strings. The binary unit for UTF-8 is 8 bit. The binary unit for UCS-2 is 16-bit, i.e., wide character.

Declaring Encodings

The database code page is a PSQL database property that declares the encoding of character data stored in the database. The purpose is to help insure that character data can be interpreted correctly. However, the database code page property is just a declaration. PSQL does not validate the encoding of the data and metadata that an application inserts into a database. The application is responsible for ensuring that character data is stored and retrieved in a particular encoding. Note that the database code page only applies to text encoded with legacy code pages or UTF-8. Wide-character text is always encoded using UCS-2. A proper setting is required if the engine will be converting between wide-character text and byte-string text. The default value of the database code page is the system code page of the OS where the engine is running.

The PSQL SQL access methods infer a client code page for byte strings exchanged between the application and the access method. (Wide-character strings are always encoded with UCS-2.) On Windows, the access method assumes that the application is respecting the ACP (Active Code Page) for byte strings. On Linux, the access method assumes that the application is respecting the encoding of the locale, which is usually given by the LANG environment variable.
PSQL provides methods to ensure compatible encoding between the database engine and clients. For example, an application can specify that it wants the PSQL SQL client to translate data automatically between the database code page and the client application. This is referred to as automatic translation. Note, however, that automatic translation can translate characters only if they are present in the character sets of both code pages (the code page on the server machine and the code page on the client machine).

For backward compatibility, automatic translation in access methods is disabled by default. The application must configure the access method to enable automatic translation. When possible, the recommended method is to set the database code page and configure the access method to read and use that value.

**Collation and Sorting**

Collation is the general term for the process and function of determining the sorting order of strings of characters. Collation varies by language, so it is not possible to arrange the encoding in multilanguage code pages so that simple binary string comparison produces the desired collation order for every language. If multilevel sorting is a requirement, separate data tables are required to define the correct sorting order.
Choosing a Character Set and Encoding

To implement a globalization strategy you typically begin by identifying the character set required based on the languages or other text and character requirements you need to satisfy. The next step is to choose the encodings that support the character set. The encoding used may even be different for the database and for client applications. Let’s look at some examples.

The most global character set is the Unicode character set. Even if legacy character sets are used in the clients, they can all be translated to Unicode for storage in PSQL. For a new application or a new module, storing text in UCS-2 or UTF-8 encoding is a simple approach. However, not all applications are new.

Another consideration for applications is the technology of the client programs. If the application uses the .NET framework, the Java VM, or the UNICODE option with C/C++, the application is already processing text using wide-character strings. In these situations the main consideration is configuring PSQL to preserve that text and choosing how to store it.

If the application is using byte strings in C/C++ and the legacy PSQL ODBC driver, there are two possible paths to globalization. One is to port the application to use wide-character strings; the other is to let the application continue to support the legacy code page of the client where it is installed and to arrange for translation to Unicode storage.

A very conservative approach for existing applications is to continue using your current legacy code page and take advantage of the other languages that it supports. For example, an application developed for the English-speaking market on Windows using ANSI code page 1252 or OEM code page 850 can also support Western European languages without any change in application storage. The main changes would be to localize program text.

Note User Data and Metadata

PSQL has two types of text that it must handle. The first is user data, which is mostly manipulated by the application, and also by index ordering and by SQL string functions. The second type is metadata, which is the names of SQL objects, such as tables, columns, and indexes. Metadata does not handle UCS-2 encoding, and so follows the legacy code page of the database code page declaration. SQL queries can contain both user data in string literals, and metadata in object names. Thus when discussing SQL queries, we must distinguish the character sets of user data and metadata, even when we are using one of the Unicode encodings for the SQL text as a whole.

PSQL is not prepared to handle mixed encodings in text storage. The application should consider such text to be BINARY storage and handle all encoding translations in the application. PSQL assumes that all CHAR type data and SQL metadata respect the database code page, and that all NCHAR type data is UCS-2.

The following sections cover some specific storage cases, Multilingual Database Support With Unicode UTF-8 and Multilingual Database Support With Unicode UCS-2. Following is a section on handling legacy OEM code pages, Multilingual Database Support with Legacy and OEM Encodings.
Multilingual Database Support With Unicode UTF-8

If you choose to store text as UTF-8 you will continue to use the CHAR, VARCHAR, and LONGVARCHAR relational types. You also need to consider such aspects as the Unicode support for the operating system on which your application runs, the string manipulation libraries available to your application, the PSQL access method(s) your application uses, any columns that may need a different data type, and so forth.

When to Use Unicode UTF-8

Unicode UTF-8 encoding is a good choice for the following:

- You want to add new language support to an existing application but keep application changes fairly minimal. For example, you have a PSQL database with ANSI-only characters (English, for instance). You want to extend your application to include data in English, German, Polish, and Czech. UTF-8 provides compact storage requirements for European scripts because it requires, at most, two bytes per character.
- A Web application since many Web platforms use UTF-8. Because Unicode UTF-8 is ASCII-compatible and compact for Latin-based language character sets, it is often used as a standard encoding for interchange of Unicode text.
- A Linux application because Linux distributions well support UTF-8 string handling.

Unicode UTF-8 Support in PSQL

One of the code pages supported by PSQL is UTF-8. For UTF-8 text storage, you would set the DB code page for your PSQL database to UTF-8.

Note that with UTF-8, string storage is byte strings. For byte strings, PSQL provides the relational data types CHAR, VARCHAR, and LONGVARCHAR, and the Btrieve data types STRING and ZSTRING. See also Data Types in SQL Engine Reference. Columns will likely be wider when storing UTF-8 because European languages often require two bytes per character instead of a single byte for legacy code pages.

All string data inserted by your application for existing CHAR, VARCHAR and LONGVARCHAR data types are interpreted as UTF-8 strings. You can configure the PSQL SQL access methods to automatically translate to UTF-8 (see Access Methods for Unicode UTF-8 Support).

When the database code page is UTF-8 and the client environment supports Unicode (wide-character or UTF-8), SQL text supports Unicode characters in CHAR literals. With any other database code page, general Unicode characters must be in NCHAR literals.

Collation and Sorting

PSQL supports only code point order for collation and sorting with UTF-8 storage.

Access Methods for Unicode UTF-8 Support

The PSQL access methods ODBC, JDBC, and ADO.NET support translation to UTF-8 storage. These access methods exchange text values with the application as UCS-2 wide-character strings or as legacy byte strings for the ANSI ODBC drivers. When properly configured, the access methods translate the application text values to UTF-8 for transmission to the storage engine.
If your application uses the ANSI ODBC driver on Windows, all data will be converted by the
Windows Driver Manager to the client legacy code page for byte strings. This results in the loss of
any characters that are not in the legacy character set. You may also need to convert your application
to use the Unicode ODBC driver.

If your application uses the ANSI ODBC driver on Linux, you should set the app locale to use UTF-
8 as the string encoding. For completeness, also declare pvtranslate=auto in the connection string
and declare the database code page to be UTF-8.

For JDBC, your application needs to specify pvtranslate=auto in the connection string to the JDBC
driver. See Connection String Overview in JDBC Driver Guide.

For ADO.NET, your application needs to specify pvtranslate=auto in the connection string to the
database engine. See Adding Connections in Data Provider for .NET Guide.

Migrating an Existing Database to Unicode UTF-8

All text data must be converted from any legacy code page to UTF-8. Columns will likely need to be
widened to accommodate the longer UTF-8 byte strings. Any non-ASCII metadata, such as table names,
must be converted from the legacy code page to UTF-8. Given these combined changes, it is reasonable
to migrate the database by copying from the old schema, using the legacy code page, to the new schema
with UTF-8 as the database code page.

Note In the special case where all existing data and metadata is pure ASCII, it is possible to just
change the database code page to UTF-8.

All existing (7-bit) ASCII byte strings are also valid UTF-8 byte strings.
Multilingual Database Support With Unicode UCS-2

If you choose to store text as UCS-2 you will use the NCHAR, NVARCHAR, and NLONGVARCHAR relational types. This has no effect on your ability to also store other text in the CHAR family of relational types. You also need to consider such aspects as the Unicode support for the operating system on which your application runs, the string manipulation libraries available to your application, the PSQL access method(s) your application uses, any columns that may need a different data type, and so forth.

When to Use Unicode UCS-2

Unicode UCS-2 is a good option for the following situations:

- Your application supports Asian character data. With UCS-2, all characters are stored as two bytes which provides more compact storage than UTF-8 for Asian character data.
- Your application will not be globalizing all of the storage and so will have wide-character columns together with legacy byte string columns.
- Your application needs better compatibility for wide character data with Java, ADO.NET, or wide-character clients. Such applications use UCS-2 for application strings.

Unicode UCS-2 Support in PSQL

Wide-character storage is a separate type from byte-string storage and may be used alongside byte-string storage.

For wide character strings, PSQL provides the relational data types NCHAR, NVARCHAR, and NLONGVARCHAR, and the Btrieve data types WSTRING and WZSTRING. All data inserted by your application for NCHAR, NVARCHAR, NLONGVARCHAR, WSTRING and WZSTRING data types is interpreted as wide character strings.

PSQL supports Unicode characters in NCHAR literals in SQL query text. Text data in CHAR literals is translated to the database code page. Keep in mind that database code pages other than UTF-8 cannot handle most Unicode characters.

Collation and Sorting with Unicode UCS-2

PSQL supports only code point order for collation and sorting with UCS-2 storage.

Access Methods for Unicode UCS-2 Support

Using NCHAR storage with the SQL access methods requires that applications specify the appropriate NCHAR SQL types to the access method. Use of CHAR types can cause conversion from NCHAR to CHAR and loss of data if the database code page is not UTF-8.

- ODBC applications should use the SQL_WCHAR, SQL_WVARCHAR, and SQL_WLONGVARCHAR SQL types for parameters.
- Windows ODBC applications should be using wide-character strings, identified to ODBC as SQL_C_WCHAR. Also, Windows ODBC applications should use the PSQL Unicode ODBC driver. Use of the legacy ANSI ODBC driver will result in loss of data because the Windows Driver Manager will convert all wide-character strings to byte string.

- Linux ODBC applications should use a locale with a UTF-8 encoding and use UTF-8 in application byte strings (SQL_C_CHAR) to provide data for NCHAR values (SQL_WCHAR).

- JDBC applications should set string parameters to type NVARCHAR or use the 'N' methods such as setNString() to send string data as an NCHAR SQL type. Also, set pvtranslate=auto in the connection string and set the database code page. Otherwise, JDBC will remain backward compatible and use the declared or default byte-string encoding for string data. See Connection String Overview in JDBC Driver Guide.

- ADO.NET applications need to specify pvtranslate=auto in the connection string to the database engine. See Adding Connections in Data Provider for .NET Guide.

In all cases, use NCHAR literals in SQL text.

**Migrating an Existing Database to Unicode UCS-2**

The tasks for migrating an existing database to support a UCS-2 application involve converting byte strings to wide character strings. The extent of the conversion depends on your application. ALTER TABLE can be used to convert columns provided that the database code page is set to match the encoding for existing CHAR columns. It is only necessary to convert columns that will store globalized data. Columns that will never store characters outside of the character set of the database code page can remain as CHAR types.

If your application uses the ANSI ODBC driver on Windows, you need to convert your application to wide character data and use the PSQL Unicode ODBC driver.
Multilingual Database Support with Legacy and OEM Encodings

If you choose to store text using a legacy or OEM code page, you will use the CHAR, VARCHAR, and LONGVARCHAR relational types. You may use wide-character applications if you configure the access method to do conversion to your legacy code page.

When to Use a Legacy Code Page

Using a legacy code page is a compact and efficient way to store text data provided that the legacy code page defines a character set that meets the needs of your application.

Legacy Code Page Support in PSQL

The PSQL database engine assumes that CHAR data is encoded using the database code page. The access methods have configuration options that help ensure that this is the case.

Older applications that use OEM code pages for storage did not declare the database code page. You can continue to work with this situation so long as the application is careful to use the appropriate relational string functions.

Collation and Sorting With Legacy Code Pages

The default collation in PSQL is binary ordering of the encoded bytes. The declared database code page does not affect this default collation.

PSQL provides both ACS (Alternate Collating Sequence) and ISR (International Sorting Rules) mechanisms for controlling collation. These may be declared on Btrieve indexes and on relational columns. PSQL will always use a particular ACS for the ASCII character set when using the “CASE” declaration on a relational column or the “case insensitive” bit on a Btrieve index.

Access Methods for Legacy Code Pages

All PSQL access methods support byte-string text. Some assume that the database code page is the same as the application's code page, and some allow configuration.

- The ANSI ODBC drivers provide a DSN setting, “OEM/ANSI” translation, which declares that the application OEM code page (as distinct from ANSI code page) is the database code page. The ANSI ODBC drivers also support the “encoding” property in connection strings and the “pvtranslate=auto” option.
- The Unicode ODBC driver always operates as if the “pvtranslate_true” property is set and translates application CHAR data to the database code page as needed.
- The JDBC driver has an explicit “encoding” property in the connection string. The driver will set encoding to the database code page if the “pvtranslate=auto” property is given.
- The ADO.NET provider has an explicit “Encoding” connection property. The driver will set Encoding to the database code page if the “PvTranslate=Auto” connection property is given.
- The PDAC access method for Delphi has the OEMConversion property that controls whether CHAR data is sent to the engine using the OEM code page or the ANSI (ACP) code page.
**Migrating an Existing Database to a Different Legacy Code Page**

Changing the code page of a database requires copying the data to a new database that uses the new code page. The copy must be done via your application. PSQL does not translate code pages within a transaction.

If your database metadata (e.g., table names) use characters that require translation, those changes must be made when creating the schema of the new database.
Database Code Page and Client Encoding

As discussed in Concepts and Definitions, above, encoding specifies how character data is translated between the PSQL database engine and a PSQL client application. PSQL handles much of the complexity of the encoding between client and server and the various combinations of operating system, languages, and access method. The encoding enhancements are divided into database code page and client encoding. The two types of encoding are separate but interrelated.

Database code page and client encoding apply only to the Relational Engine. The MicroKernel Engine is not affected.

Database Code Page

The database code page is a database property that specifies the encoding used for character data stored in the database. The default database code page is “server default,” meaning the operating system (OS) code page on the server where the database engine is running. (The OS code page is generally referred to as the “OS encoding,” which is the phrase used throughout this chapter.)

Database code page is particularly handy if you need to manually copy PSQL DDFs to another platform with a different OS encoding and still have the metadata correctly interpreted by the database engine.

When you create a database with PSQL, the default is to use the active code page for the machine on which the database engine is running. For example, on a machine running Windows for English, the code page is assigned to 1252. PSQL encoding translation is set to “none.” Your application must either use the same code page as the PSQL database, or ensure that encoding translation is set to “automatic.”

Supported Code Pages

PSQL supports the following code pages. (All of the code pages listed use byte string storage.)

- ASCII
- EUCJP
- ISO8859_1
- UTF-8
- Windows Code Pages
  - CP437, CP737, CP775, CP850, CP852, CP855, CP857, CP858, CP862, CP866, CP932
  - CP1250, CP1251, CP1252, CP1253, CP1254, CP1255, CP1256, CP1257, CP1258

Client Encoding

Client encoding is the data encoding used by an application on a PSQL client. An application can manage text in any encoding it chooses. A compatible encoding must be established between the database engine and the client application.

PSQL can automatically translate between different encodings used by the database engine and clients provided the characters are present in both the code page on the server machine and the code page on the client machine.

Data translation, if required, occurs at the client. (Translation is not always required—for example, when the client operating system (OS) encoding matches the server OS encoding.)
Database Globalization

**Encoding Support in PCC**

You can use PCC to set the database code page when you create a database or to modify the code page setting for an existing database.

*Note* Changing the database code page property does not change any data in the database. However, changing the database code page for an existing database will affect how existing data entries are interpreted.

PSQL Control Center (PCC) is, itself, a client application to the database engine. As a client, PCC lets you specify the encoding to use for each database session when PCC reads and inserts metadata and data. The default for an existing database is to use the encoding of the machine where PCC is running. This is the legacy behavior of PCC. The default for a new database is to use automatic translation. See PCC Connection Encoding in PSQL User's Guide.

The following table explains the interaction between the settings for PCC connection encoding and database code page. PCC connection encoding applies only to PCC. It has no effect on other client applications.

<table>
<thead>
<tr>
<th>PCC Connection Encoding Set to a Specific Encoding</th>
<th>PCC Connection Encoding Set to &quot;Automatic Translation&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCC ignores the database code page and uses the encoding specified to read and insert CHAR data, string literals, and metadata. NCHAR data is not affected by this setting. (This is the legacy behavior of PCC.)</td>
<td>PCC and the database automatically establish the encoding for CHAR data and metadata. String literals in queries are sent to the engine as Unicode. NCHAR data is not affected by this setting.</td>
</tr>
</tbody>
</table>

**Encoding Support in Btrieve API**

When using the Btrieve API, you must provide file names and paths in the local encoding used in your application. The Btrieve API handles the differences between OS encoding on the server and client.

**Encoding Support in DTI**

When using the Distributed Tuning Interface (DTI), you must provide file names and paths in the local encoding used in your application. DTI handles the differences between OS encoding on the server and client.

If you use the DTI API to create a database, you may specify the database code page property at the time of creation. This property may be used by SQL access methods to configure automatic translation of character data.

**Encoding Support in ADO.NET**

The .NET Framework and .NET applications use UTF-16 strings. These must be translated to a code page when storing text in CHAR columns.

The connection property "PVTranslate=Auto" will set the connection Encoding to the database code page. It is also possible to set the Encoding property directly.
For more information, see Adding Connections, PsqlConnectionStringBuilder Object and Character Set Conversions in Data Provider for .NET Guide.

**Encoding Support in JDBC**

The Java Virtual Machine and Java applications use UTF-16 strings. These must be translated to a code page when storing text in CHAR columns.

The connection property “PVTranslate=Auto” will set the connection Encoding to the database code page. It is also possible to set the Encoding property directly.

When the PVTranslate=Auto property is given, the JDBC driver will send string literals to the engine as Unicode. Without this property, the legacy behavior is to translate string literals to the database code page. If your application uses NCHAR string literals (e.g. “N’ABC’”), it should set the PVTranslate=Auto connection property.

See Connection String Elements in JDBC Driver Guide.

**Encoding Support in ODBC**

The PSQL ODBC drivers support a number of mechanisms to control client encoding.

When configuring a DSN, it is possible to select the encoding options “Automatic”, “OEM/ANSI”, and “None”. The “Automatic” setting causes the driver to translate from the client encoding to the database code page. The OEM/ANSI setting causes the driver to translate from the client encoding to the corresponding OEM code page. The None setting prevents the driver from doing any text translation. See Encoding Translation in ODBC Guide for more details.

**Legacy Conversion Methods for OEM-to-ANSI Data**

If a database has OEM character data in it, a legacy solution is to specify OEM/ANSI conversion in the access method. This topic discusses some legacy methods for Linux clients using OEM character data.

> **Note** While the legacy methods are still supported, the recommendation is to specify the OEM code page for the database and have the access methods use automatic translation as discussed above.

See also OEM/ANSI Conversion in ODBC Guide.

When using ODBC, Win32 encoding is expected to be SHIFT-JIS.

Japanese versions of Linux by default have their encodings typically set to EUC-JP or UTF-8.

When using Japanese versions of Linux, a client can connect to another Linux server (for example, locally), or to a Win32 SHIFT-JIS server. It is also possible to connect to a database encoded in SHIFT-JIS but located on a Linux server.

Use the following instructions for your listed configuration. In each case, it is assumed that the application itself does not do any conversion and uses the encoding that is native for the machine.

- Connecting a Linux EUC-JP Client to a Win32 SHIFT-JIS Server
- Connecting a Linux UTF-8 Client to a Win32 SHIFT-JIS Server
- Connecting a Linux UTF-8 Client to a Linux UTF-8 Server
Connecting a Linux UTF-8 Client to a Linux EUC-JP Server


Connecting a Linux EUC-JP Client to a Win32 SHIFT-JIS Server

The server requires that everything is received as SHIFT-JIS. The client requires that the server send everything as EUC-JP.

To accomplish this, the client DSN settings in ODBC.INI (located by default in /usr/local/psql/etc) used to connect to the given database should be set up as follows:

```
[dbclient]
Driver=/usr/local/psql/lib/libodbcci.so
Description=PSQL ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/psql/lib/libxlate.so.10
TranslationOption=90000932
```

The `TranslationDLL` line specifies the translation library that the ODBC client interface should use. The `TranslationOption` line specifies that the translation needs to occur from 9000 (representing EUC-JP) to 0932 (representing SHIFT-JIS).

Using this example, all data coming from the client will be translated into SHIFT-JIS before it gets to the server, and to EUC-JP before the data is received by the client.

Connecting a Linux UTF-8 Client to a Win32 SHIFT-JIS Server

The server requires that everything is received as SHIFT-JIS. The client requires that the server send everything as UTF-8.

To accomplish this, the client DSN settings in ODBC.INI (by default in /usr/local/psql/etc) used to connect to the given database should be set up as follows:

```
[dbclient]
Driver=/usr/local/psql/lib/libodbcci.so
Description=PSQL ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/psql/lib/libxlate.so.10
TranslationOption=90010932
```

The `TranslationDLL` line specifies the translation library that the ODBC client interface should use. The `TranslationOption` line specifies that the translation needs to occur from 9001 (representing UTF-8) to 0932 (representing SHIFT-JIS).

Using this example, all data coming from the client will be translated into SHIFT-JIS before it gets to the server, and to UTF-8 before the data is received by the client.


Using this configuration, no changes to the DSN description are needed. Use the DSN as it was created by the `dsnadd` utility.
Connecting a Linux UTF-8 Client to a Linux UTF-8 Server

Using this configuration, no changes to the DSN description are needed. Use the DSN as it was created by the dsnadd utility. See dsnadd in PSQL User's Guide.

Connecting a Linux UTF-8 Client to a Linux EUC-JP Server

The server requires that everything is received as EUC-JP. The client requires that server send everything as UTF-8.

To accomplish this, the client DSN settings in ODBC.INI (by default in /usr/local/psql/etc) used to connect to the given database should be set up as follows:

```
[dbclient]
Driver=/usr/local/psql/lib/libodbcci.so
Description=PSQL ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/psql/lib/libxlate.so.10
TranslationOption=90019000
```

The TranslationDLL line specifies the translation library that the ODBC client interface should use. The TranslationOption line specifies that the translation needs to occur from 9001 (representing UTF-8) to 9000 (representing EUC-JP).

Using this example, all data coming from the client will be translated into EUC-JP before it gets to the server, and to UTF-8 before the data is received by the client.


This situation is possible if you have a SHIFT-JIS database on a Win32 engine, and you want to move all the files to the Linux EUC-JP server. In this case, the database resides on a EUC-JP Linux machine, but all the data inside the DDF files and data files are in SHIFT-JIS.

In this case, your DSN should be set up as follows:

```
[dbclient]
Driver=/usr/local/psql/lib/libodbcci.so
Description=PSQL ODBC Client Interface: JPN-2000SERVER:1583/dbclient
ServerDSN=DEMODATA
ServerName=JPN-2000SERVER:1583
TranslationDLL=/usr/local/psql/lib/libxlate.so.10
TranslationOption=90000932
CodePageConvert=932
```

The last line specifies that even though the server uses EUC-JP encoding, it should treat the data on the server as SHIFT-JIS.

Encoding Support for Wide ODBC Driver

PSQL supports UCS-2 with ODBC with a driver for wide character data and defaults for DSN encoding translation. See Encoding Translation and ODBC Connection Strings in ODBC Guide.
ODBC Driver for Applications with Wide Character Data

PSQL provides an ODBC driver for 32-bit and 64-bit applications that use wide character data. The driver is for Windows operating systems only and is an addition to the previous set of drivers.

Table 16  PSQL ODBC Driver for Wide Character Data

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSQL ODBC Unicode Interface</td>
<td>• Connects to a local or remote named database.</td>
</tr>
<tr>
<td></td>
<td>• With the 32-bit ODBC Administrator, creates 32-bit DSNs for use by 32-bit</td>
</tr>
<tr>
<td></td>
<td>applications that use wide character data. The 32-bit driver is installed</td>
</tr>
<tr>
<td></td>
<td>with all PSQL editions.</td>
</tr>
<tr>
<td></td>
<td>• With the 64-bit ODBC Administrator, creates 64-bit DSNs for use by 64-bit</td>
</tr>
<tr>
<td></td>
<td>applications that use wide character data. The 64-bit driver is installed</td>
</tr>
<tr>
<td></td>
<td>with all PSQL editions when installing on a 64-bit platform.</td>
</tr>
</tbody>
</table>

On Linux, the system encoding is usually UTF-8, which allows SQL text to contain any Unicode character code point. The PSQL ODBC Unicode Interface driver is not available on Linux because an application can use the PSQL ODBC Client Interface driver with UTF-8. A Linux application can handle wide character data either as UTF-16 strings (SQL_C_WCHAR) or request conversion to the system encoding (usually UTF-8) as SQL_C_CHAR. SQL text using UTF-8 is compatible with the existing Pervasive ODBC Client Interface driver so an additional ODBC driver on Linux is not required.

Default for DSN Encoding Translation

The encoding translation options for a DSN specify how character data is translated between the PSQL database engine and a PSQL client application that uses ODBC. The default for encoding translation depends on the PSQL ODBC driver that you use.

Table 17  DSN Encoding Translation Default

<table>
<thead>
<tr>
<th>Driver Name</th>
<th>Encoding Translation Default</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSQL ODBC Unicode Interface</td>
<td>Automatic</td>
<td>The connection string parameter PvTranslate also defaults to &quot;auto.&quot;</td>
</tr>
<tr>
<td>PSQL ODBC Interface</td>
<td>None</td>
<td>Same default as with the previous version of PSQL.</td>
</tr>
<tr>
<td>PSQL ODBC Client Interface</td>
<td>None</td>
<td>Same default as with the previous version of PSQL.</td>
</tr>
<tr>
<td>PSQL ODBC Engine Interface</td>
<td>None</td>
<td>Same default as with the previous version of PSQL.</td>
</tr>
</tbody>
</table>
The ODBC drivers process SQL text differently depending on the driver and the setting for the DSN encoding translation.

Table 18  PSQL ODBC Driver and DSN Encoding Translation Setting Effect on SQL Text

<table>
<thead>
<tr>
<th>Setting</th>
<th>Processing of Incoming SQL Text</th>
<th>PSQL Driver</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>ODBC Unicode Interface</td>
</tr>
<tr>
<td>Automatic</td>
<td>SQL text gets converted to UTF-8 then sent to the database engine. The code pages for Client, Server, and database are ignored.</td>
<td>Yes&lt;sup&gt;1&lt;/sup&gt;</td>
</tr>
<tr>
<td></td>
<td>SQL text gets converted to the database code page then sent to the database engine.</td>
<td>No</td>
</tr>
<tr>
<td>None</td>
<td>SQL text is not translated between the Client and database engine.</td>
<td>Yes</td>
</tr>
<tr>
<td>OEM/ANSI</td>
<td>SQL text in the Client code page is converted to the OEM/ANSI encoding and then sent to the database engine.</td>
<td>Yes&lt;sup&gt;3&lt;/sup&gt;</td>
</tr>
</tbody>
</table>

<sup>1</sup> With the encoding translation set to “Automatic,” you can use NCHAR columns and NCHAR literals with wide character data.

<sup>2</sup> The assumption is that the Client and database engine use the same operating system encoding.

<sup>3</sup> If the SQL text is wide character, it is first converted to the Client encoding. If the SQL text is not wide character, it is already in the Client encoding. The SQL text is then converted to the OEM encoding and sent to the database engine.
Unicity Support in PSQL Utilities

*Unicode Support in PSQL Control Center (PCC)*

See also, the section Encoding Support in PCC earlier in this chapter.

**Dialogs for Opening and Saving Files**

The PCC dialogs for opening and saving SQL documents, saving exported schemas, and importing and exporting table data have all been enhanced to accommodate a variety of file encodings. Previously, these files were presumed to be in the default system code page. It is not possible to select a number of Unicode encodings when saving files. When opening a file, the new dialogs detect whether the file uses a byte order mark (BOM) to identify the Unicode encoding. The opening dialogs also allow you to set the expected encoding of the file. For your convenience, a new PCC setting controls the default encoding used in these dialogs.

For more information on these new features, see PSQL User's Guide under the topics Dialogs for File Open and File Save, Wide Character Data Support for Import Data, Export Data, and Export Schema, and File Encoding Preferences.

**Bulk Data Utility (BDU)**

The Bulk Data Utility (BDU) is a command line utility that allows you to load data from a delimited text file into a PSQL table. A command line parameter, `-c encoding`, is provided to specify the data encoding to use when loading the data file. The encoding options are UTF-8, UTF-16LE, and UTF-16BE. If a data file contains a byte order mark (BOM), BDU uses the encoding specified by the BOM. That is, if a data file uses a BOM to indicate an encoding of UTF-8, UTF-16LE, or UTF-16BE, BDU uses that encoding regardless of what value you specify for the encoding parameter on the command line. Without a BOM or the `-c` parameter, BDU defaults to using the system code page.

See `bdu` in PSQL User's Guide.
Support for Collation and Sorting

What is Collation and Sorting?
Collation refers to the ordering of the characters in a character set. For example, one collation might put digits before letters and another might put them after. Sorting is the rearrangement of a set of data so that the text is in collation order.

PSQL supports the specification of a named collation on byte-string text segments. Indexes will sort the record keys according to the specified collation.

Sort Order with no Collation Sequence Specified
When no collation is specified, PSQL will use a default collation that orders characters according to the value of their encoding. Thus the relative ordering of two characters depends on which code page is in use. The default is ascending order, i.e., from lowest code page value to highest. You can optionally set this to descending order. See Sort Order in PSQL Programmer’s Guide for more information.

Collation Support in Wide-char columns
PSQL supports the default collation of wide-character data according to code point value. Both UCS-2 and UTF-8 text will sort in code point order.

Collation Support using an Alternate Collating Sequence (ACS)
You can specify an alternative to the default code page collation order. This user-defined Alternate Collating Sequence or ACS is a mapping between the code page collation order and the desired collation order. You can define one or more alternate sequences for determining the collation of string keys of type STRING, LSTRING, and ZSTRING. For example, you can use a user-defined ACS to specify a collation that places numbers after letters or changes the ordering of upper and lower case letters. PSQL comes with an ACS, named UPPER.ALT, that maps the lower case letters to sort as equivalent to uppercase letters. (This could also be achieved by setting case insensitivity but shows what can be done with an ACS.)

Essentially, the user-defined ACS is a table that associates the code page sequence position for a character with the alternate desired sequence position. Creating an ACS is described in Alternate Collating Sequences in PSQL Programmer’s Guide. Examples are provided there, also. You specify the ACS for key value fields in the definition of the layout of the data file (see Specifying a Key’s Alternate Collating Sequence (in this guide) and Data Layout in PSQL Programmer’s Guide).

For additional information about setting an ACS, see Create (14), Create Index (31) and Get Next Extended (36) in Btrieve API Guide, Alternate Collating Sequence (ACS) Files in DDF Builder User’s Guide and SET DEFAULTCOLLATE in SQL Engine Reference.

Collation Support using an International Sort Rule (ISR)
Another type of ACS is an International Sort Rule or ISR. An ISR is a pre-defined alternate collating sequence for language specific sort orders. You can use an ISR to correctly sort languages such as German with the letters ä, ö, ü (sorted as ae, oe, ue) and ß (sorted as ss). PSQL provides a number of ISR tables in the COLLATE.CFG file in your PSQL installation. Examples of their use can be found in Sample Collations Using International Sorting Rules in PSQL Programmer’s Guide. See the references for Alternate Collating Sequences, above, for more information.
Database Globalization
Locale Support

An important aspect of globalization is locale, which is a model and definition of a native-language environment. A locale consists of a number of categories for which country-dependent formatting or other specifications exist. For example, a locale defines date and time formatting conventions, monetary conventions, decimal formatting conventions, and collation (sort) order. Depending on the operating system, a locale may be called a region.

More than one locale can be associated with a particular language, which allows for regional differences. For example, English can have a United States locale and a Great Britain locale.

When executing string functions, PSQL uses the locale of the operating system on which the database engine is running. PSQL uses the locale of the Client when converting data types as requested by the application through one of the PSQL access methods.

For more information, see SET DECIMALSEPARATORCOMMA, Comma as Decimal Separator and SET TIME ZONE in SQL Engine Reference.
Database Globalization
Setting Up Referential Integrity

An Introduction to Referential Integrity Structures

Referential Integrity is a system of checks and balances that you can create in your database to ensure that tables with related data remain synchronized.

- Concepts of Referential Integrity
- Setting up Primary Keys
- Setting up Foreign Keys
- Interactions Between Btrieve and Relational Constraints
Setting Up Referential Integrity

Concepts of Referential Integrity

Referential Integrity (RI) allows you to modify or prohibit updates, inserts, or deletes based on whether identical field values exist in the same or other tables.

Definitions

A good understanding of RI depends upon a clear understanding of several important terms:

Rule

A rule is a simple statement of cause and effect, carried out by the RI system defined in the database.

Example A

For example, a delete rule defines what happens to records containing a foreign key when a record containing a primary key is deleted: “When the record containing ‘Bhargava Building’ is deleted, all rows in Table A that reference that record are deleted.”

A delete rule can also prohibit the row containing the primary key value from being deleted if there are any foreign key values that reference the given primary key value.

Example B

An update rule defines what happens to a record containing a foreign key when a user attempts to update the record or add a new record: “When a user attempts to insert a new record to Table B, reject the attempt if the building name does not exist in Table C.”

Primary key

A primary key is the column or columns upon which a rule depends. Only one primary key is permitted in any table, and the primary key must not allow duplicate values. For an update rule, the primary key is the column or columns against which updated or inserted columns are compared to determine if the updated or inserted record should be allowed.

In Example A, the column containing “Bhargava Building” is the primary key.

In Example B, the column in Table C that contains the building name is the primary key.

Foreign key

A foreign key is the column or columns that are compared against a primary key to determine how to proceed.

In Example A above, the column in Table A that may contain the value “Bhargava Building” is the foreign key.

In Example B above, the column in Table B that contains the building name is the foreign key.

Cascade

A cascade rule is a rule in which the database permits the desired operation to occur, then enforces RI by changing other tables or rows to synchronize with the first operation. For example, if a delete cascade rule is defined, deleting a record in the primary key table causes the database to find and delete all rows
throughout the database that have foreign key values the same as the primary key value of the deleted row.

Restrict

A restrict rule is a rule in which the database decides whether or not to permit the desired operation based on existing values in the database. For example, if an update restrict rule is defined, an attempt to add a row to a table containing a foreign key causes the database engine to compare the value in the foreign key field to the values in the primary key. If there is no primary key row with the same value, the new row is not permitted to be added to the foreign key table.

Understanding Keys and Rules

This section explores the concepts behind primary keys and foreign keys in further detail.

Table 19 Primary and Foreign Keys

<table>
<thead>
<tr>
<th>Table A</th>
<th>Table B</th>
</tr>
</thead>
<tbody>
<tr>
<td>student_ID</td>
<td>Name</td>
</tr>
<tr>
<td>20543</td>
<td>John</td>
</tr>
<tr>
<td>20577</td>
<td>Mary</td>
</tr>
<tr>
<td>20543</td>
<td>ENG-101</td>
</tr>
<tr>
<td>20543</td>
<td>AST-202</td>
</tr>
</tbody>
</table>

In the example shown above, the column named student_ID in Table A (A.student_ID) is an IDENTITY data type that does not allow two rows to have the same value. Every student has a unique ID number. We will define student_ID as the primary key of Table A.

We can then define the column named stud_ID in Table B (B.stud_ID) as a foreign key that references A.student_ID. Note that the data type of stud_ID must be a type that can be compared with IDENTITY, such as INTEGER. The data types of primary and foreign keys must be compatible. You can have as many foreign keys as you need in order to enforce your desired referential integrity scheme. Multiple foreign keys can reference the same primary key.

The table with the primary key can be referred to as the parent table, while the table with the foreign key is called the child table. Once the keys are defined, we have a range of behaviors to choose from, as shown in Table 20. You can define as many rules as fit your needs, but you can only have one of each type. For example, if you define a delete restrict rule, you cannot define a delete cascade rule on the same keys, because the two behaviors are mutually exclusive.

Table 20 Choices for RI Rules

<table>
<thead>
<tr>
<th>If you want this behavior...</th>
<th>... define this rule:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Do not allow a row to be inserted or updated in Table B unless the proposed value of B.stud_ID matches any value in A.student_ID.</td>
<td>Update Restrict</td>
</tr>
<tr>
<td>Do not allow a row to be deleted from Table A if any value of B.stud_ID matches that row.</td>
<td>Delete Restrict</td>
</tr>
<tr>
<td>If a row is deleted from Table A, delete all rows from Table B in which B.stud_ID matches the value of A.student_ID in the deleted row.</td>
<td>Delete Cascade</td>
</tr>
</tbody>
</table>
Update Restrict
Continuing with the example, setting an update restrict rule ensures that the value of $B.stud_ID$ in any new or updated row must first exist in $A.student_ID$. It follows, then, that you must have rows in Table $A$ before you can add any rows in Table $B$. Stated another way, you must create at least one parent row before you can create a child row.

Delete Restrict
In the example, setting a delete restrict rule ensures that a row from Table $A$ cannot be deleted if any rows in Table $B$ reference that row. You cannot delete the row with Name value “John” because John’s student ID is referenced in Table $B$.

Once all rows from Table $B$ that reference John’s student ID are deleted, then John’s row can be deleted from Table $A$.

Delete Cascade
In the example, setting a delete cascade rule ensures that both records in Table $B$ are deleted if the row with Name value “John” is deleted.

PSQL allows a circular delete cascade on a table that references itself. Because of this, use delete cascade with caution. Ensure that you do not inadvertently delete all records in the parent table, the child table, or both.

An example helps clarify how such cascading deletion could occur. Suppose that you create the following table, $d_3$, with two columns:

```
CREATE TABLE d3 (c1 INT PRIMARY KEY, c2 INT)
INSERT INTO d3 VALUES (2,2)
INSERT INTO d3 VALUES (3,2)
INSERT INTO d3 VALUES (1,3)
INSERT INTO d3 VALUES (4,1)
```

You then alter the table to add a foreign key with a delete cascade rule:
```
ALTER TABLE d3 ADD FOREIGN KEY (c2) REFERENCES d3 ON DELETE CASCASE
```

The following statement deletes all rows in the table:
```
DELETE FROM d3 WHERE c1 = 2
```

Why are all rows deleted instead of just the row where $c1 = 2$?

Delete cascade deletes any row with a foreign key equal to the primary key that is deleted. The second row has a foreign key relationship to the first row. Similarly, the third row has a foreign key relationship with the third, and the fourth row with the third. Because of the foreign key relationships, the delete cascade rule traversed all of the rows, causing the second row to be deleted because of the first, the third because of the second, and the fourth because of the third.

PSQL does not allow circular delete cascade on tables that reference each other. For example, consider the following scenario in which you have tables $d_1$ and $d_2$:

```
CREATE TABLE d1 (c1 INT PRIMARY KEY, c2 INT)
CREATE TABLE d2 (e1 INT PRIMARY KEY, e2 INT)
```

The following alter statement is allowed:
```
ALTER TABLE d1 ADD FOREIGN KEY (c2) REFERENCES d2 ON DELETE CASCASE
```
The following alter statement is not allowed because tables d1 and d2 already have a delete cascade relationship:

```
ALTER TABLE d2 ADD FOREIGN KEY (e2) REFERENCES d1 ON DELETE CASCADE
```
Setting up Primary Keys

You can create primary keys using SQL statements or PSQL Control Center. See Column Tasks in PSQL User's Guide.

Creating a Primary Key During Table Creation

You can create a primary key when you create a table, by using the PRIMARY KEY keywords in your CREATE TABLE statement. A primary key can consist of one or more columns. The following example shows the column named `id` being created then being designated the primary key:

```
CREATE TABLE mytable (id INTEGER,
                      myname CHAR(20),
                      PRIMARY KEY(id))
```

The next example shows how to create a primary key using more than one column as the unique key value:

```
CREATE TABLE mytable (id INTEGER,
                      myname CHAR(20),
                      PRIMARY KEY(id, myname))
```

Regardless of whether you specify the UNIQUE attribute on the column or columns that you designate as a primary key, the database engine automatically creates an index on the designated columns that does not allow duplicate values or null values in the columns. Null values are never allowed in a key column. Every primary key value must be unique.

For more examples, see CREATE TABLE in SQL Engine Reference.

Adding a Primary Key to an Existing Table

You can add a primary key to an existing table through PCC or by using the ALTER TABLE statement with ADD PRIMARY KEY. In PSQL User's Guide, see To set or remove a column as a primary key and SQL Editor.

You must create the primary key on a column or columns that do not allow duplicate values or null values.
If necessary, you can modify the column attributes and make the column the primary key at the same time. Here is an example using SQL:

```
ALTER TABLE mytable MODIFY id INTEGER UNIQUE NOT NULL PRIMARY KEY
```

If you want to add a primary key consisting of more than one column, you must add the key separately:

```
ALTER TABLE mytable ADD PRIMARY KEY(id, myname)
```

For more examples, see `ALTER TABLE` in SQL Engine Reference.
Setting up Foreign Keys

You can create foreign keys using SQL statements or PSQL Control Center. When you create a foreign key, you may define an associated rule at the same time. You can define multiple rules on the same key. If you create a foreign key without specifying associated rules, the default referential integrity is restrict for both update and delete.

Creating a Foreign Key During Table Creation

You can create a foreign key when you create a table, by using the REFERENCES keyword in your column definition. A foreign key can consist of one or more columns. The data types of the column(s) must be the same as the primary key that this foreign key references. The example next shows the column named your_id being created then being designated the foreign key, referencing mytable.id:

```
CREATE TABLE yourtable (your_id INTEGER REFERENCES mytable(id) ON DELETE CASCADE,
yourname CHAR(20))
```

You can also add the foreign key designation at the end of the statement. You must use this technique if you wish to use multiple columns in the key:

```
CREATE TABLE yourtable (your_id INTEGER,
yourname CHAR(20),
FOREIGN KEY(your_id, yourname) REFERENCES mytable(id, myname) ON DELETE CASCADE)
```

When you create a foreign key, the database engine adds an index on the designated columns.

For more examples, see CREATE TABLE in SQL Engine Reference.

Adding a Foreign Key to an Existing Table

You can add a foreign key to an existing table with PCC or by using the ALTER TABLE statement with ADD FOREIGN KEY. In PSQL User's Guide, see Foreign Keys Tasks and SQL Editor.

In the following example, two rules are defined for this foreign key, both a delete rule and an update rule:

```
ALTER TABLE yourtable ADD FOREIGN KEY (your_id,yourname) REFERENCES mytable(id, myname) ON DELETE CASCADE ON UPDATE RESTRICT
```

Use DELETE CASCADE with caution. See examples in Delete Cascade.

For more examples, see ALTER TABLE in SQL Engine Reference.
Interactions Between Btrieve and Relational Constraints

While PSQL is designed to support concurrent access to the same data through both the Relational Engine and the MicroKernel Engine, some features of the relational (SQL) database architecture may interfere with Btrieve access to the data. For example, features that are designed to limit relational access, such as Referential Integrity, may also limit Btrieve access in the interest of preserving data integrity.

You should fully understand Integrity Enforcement, Bound Databases, ODBC/SQL Security, Triggers, Referential Integrity and Owner Names before implementing these features on a database that is used by a transactional (Btrieve) application. In most cases you can get “Best of Both Worlds” access to your database, but since Security, Referential Integrity, and Triggers can put constraints on access or operations, some Btrieve operations may be restricted or prevented depending on the implementation.

In some cases using Integrity Enforced, Bound Databases, Security, Triggers, or Referential Integrity may cause Btrieve access to the data and/or file to be restricted or prevented when Btrieve access would violate the bounds and restrictions placed on that data. Triggers and RI mainly limit the ability to manipulate data via the Btrieve API.

Security and owner names can limit access and/or the ability to manipulate that data without the proper account, rights, and password. There are many possible combinations of these features, so only the most common ones are listed here.

Table 21 Interactions Between Relational Restrictions and Btrieve Access

<table>
<thead>
<tr>
<th>CONDITIONS</th>
<th>RESULTS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Btrieve Access allowed?</td>
</tr>
<tr>
<td>No</td>
<td>-</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

(1) Regardless of the Bound Database setting for a database, the database engine automatically stamps a data file as bound if it has a trigger, a foreign key, or a primary key that is referenced by a foreign key. For more information on the meaning of a bound database or file, see Bound Databases.
(2) Adding triggers on a table prevents access to that file from the Btrieve API, for any operations that would cause the trigger to execute. Because triggers do not react to database operations coming through the Btrieve interface, this lock-out behavior preserves the consistency of the data. See Bound Database versus Integrity Enforced for more information.

(3) When a database or file is secured, access is allowed as long as the user has permissions (that is, a relational user name and password or a valid Btrieve owner name) to that file. Files that are in a secure database but do not have Btrieve owner names set are accessible to Btrieve users. When relational security is first set up for a file that already has a Btrieve owner name, the Master user must grant relational permissions to users using the file's Btrieve owner name. See PSQL Security for more information.

(4) If a table contains referential integrity constraints, and Integrity Enforced is turned on for the given database, both Btrieve and SQL operations that would violate the constraints are disallowed. This mechanism preserves the integrity of the data regardless of the method of access.

Bound Database versus Integrity Enforced

If you do not specify the attribute “Integrity Enforced” for a named database, the database engine does not enforce any referential integrity, triggers, or security rules. If you specify the attribute “Integrity Enforced” for a named database, the MicroKernel enforces the database's defined security, RI, and triggers, regardless of the method you use to access the data. The MicroKernel enforces these rules as follows:

- Btrieve users are not subject to relational security. If you have owner names on the files, they remain in effect. If you do not have owner names on the files, any Btrieve user can access the data regardless of relational security constraints. Btrieve operations are subject to all the RI and other constraints defined in the database as well as the Trigger restrictions listed below.

- If constraints exist on a given file, Btrieve access is permitted as follows:

<table>
<thead>
<tr>
<th>Constraint on File</th>
<th>Level of Access Permitted Using Btrieve</th>
</tr>
</thead>
<tbody>
<tr>
<td>RI constraints defined</td>
<td>User can access the data and perform any operations within RI constraints.</td>
</tr>
<tr>
<td>INSERT triggers defined</td>
<td>Read-only, update, and delete operations permitted.</td>
</tr>
<tr>
<td>UPDATE triggers defined</td>
<td>Read-only, insert, and delete operations permitted.</td>
</tr>
<tr>
<td>DELETE triggers defined</td>
<td>Read-only, update, and insert operations permitted.</td>
</tr>
</tbody>
</table>

If more than one constraint exists on the bound file, the access level follows the most restrictive constraint or combination of constraints. For example, if a file has both INSERT and UPDATE triggers defined, then Btrieve users have only read-only and delete access.

“Integrity Enforced” is not directly related to “Bound Database.” A database can be bound without enforced integrity, or a database can have integrity enforced without being bound.

Bound Databases

If you specify the attribute “Bound” for a named database, the DDFs and data files in that database cannot be associated with any other database. Also, a bound data file cannot be associated with more than one table definition in the database. When you add new tables or DDFs to a bound database, the
database engine automatically binds the new objects. This behavior prevents conflicts that could cause unpredictable behavior or data integrity corruption. For example, if you used the same data file for two different table definitions in the same database, you could define RI rules on one table but not on the other. In this case, inserting a row into the table without the RI rules would violate the RI rules on the other table. Binding the data files and DDFs prevents such conflicts.

DDFs and data files can be individually bound. The database engine automatically marks a data file as bound if it has a trigger, has a foreign key, or has a primary key that is referenced by a foreign key. These files cannot be shared with another database or associated with more than one table definition.

Whether a data file is bound has no direct affect on Btrieve access to that data file. However, files that are bound often have other constraints that may limit Btrieve access.

See Also
For information on how to manipulate the “Integrity Enforced” and “Bound Database” settings for a given database, see New Database GUI Reference.
Setting Up Referential Integrity
PSQL Security

Concepts and Tasks Related to Security for the Database Engine

PSQL provides support for Btrieve owner names as well as a full implementation of database security that can be used by either Btrieve or SQL-based applications. This chapter explains the relationship between the two, and how to work with both.

- Security Models and Concepts
- Planning Security for the MicroKernel Engine
- MicroKernel Engine Security Quick Start
- Security Tasks
- Data Encryption
Database Security

By default, database security is turned off when you create a new database with PSQL v12. You turn it on by supplying a password for the “Master” user. See table Identifier Restrictions by Identifier Type for the restrictions pertaining to passwords.

Master User

Database security is based on the existence of a default user named “Master” who has full access to the database when security is turned on. By default, no password is set for the Master user. Security is enabled, or turned on, once you specify a password for the Master user.

The Master user can create groups and other users and define sets of data access permissions for these groups and users. You can add users and groups by executing SQL statements or by using PSQL Control Center (PCC).

The PUBLIC Special Group

If you want to grant the same permissions to all users, you can grant them to a special group named “PUBLIC.” The database engine automatically creates the special group PUBLIC when you turn on security. Initially, no permissions are assigned to PUBLIC.

PUBLIC is a special group because it provides default permissions for all users and groups. The database engine always checks permissions assigned to PUBLIC first. A couple of examples help clarify how PUBLIC permissions apply.

Suppose in PCC that you assign the CREATE TABLE permission to PUBLIC. You then create a user named “myuser” whose permissions in PCC do not include individual rights to create a table. Myuser can create a table because the database engine first checks default permissions in PUBLIC, and PUBLIC, in this case, grants rights to create a table.

Conversely, if a permission is not granted to PUBLIC, then the permission granted to the individual user or group applies. For example, suppose in PCC that you do not assign the CREATE TABLE permission to PUBLIC. No user can create a table unless the permissions for the user, or the group to which the user belongs, allow creating a table.

Users and Groups

After you turn on database security, you can then define groups and users. Nodes for Groups and Users appear in PSQL Explorer in PCC. See User and Group Tasks in PSQL User’s Guide.

Restrictions

- A given user cannot be a member of more than one group.
- All users in a group have exactly the permissions defined for that group. You cannot grant or revoke individual permissions for a user who is a member of a group.
- You cannot add a group to another group.
Security Models and Concepts

This section details the available security models for the MicroKernel Engine and Relational Engine. Both engines share the same level of granularity in choosing how rights are assigned to users and both support database security. The MicroKernel Engine has additional security policies that can determine how access is granted. The Relational Engine supports column-level security.

Note Specifying a security policy for the MicroKernel Engine has no effect on Relational Engine security. For purposes of discussion, however, you can think of the Database policy that is discussed under MicroKernel Engine security as the same type of security for the Relational Engine. A key difference between security for the two interfaces is that you cannot change security policies for the Relational Engine. Security is either on or off. If on, security is analogous to that of the Database policy type for the MicroKernel Engine.

Available Model for the Relational Engine

For the Relational Engine, security is either turned on or off (see Note above). By default, security is turned off. Security is turned on by supplying a password for the Master user.

With security on, you need to define, at a minimum, users through PCC who are authorized to log in to the database. For each user, you may set permissions for certain objects. In addition, you may define groups of users and set object permissions for each group.

In SQL Engine Reference, the following content applies to security for the Relational Engine:

- Permissions on Views and Stored Procedures
- ALTER GROUP
- ALTER USER
- CREATE GROUP
- CREATE USER
- DROP GROUP
- DROP USER
- GRANT
- REVOKE
- SET PASSWORD
- SET SECURITY
- psp_groups
- psp_procedure_rights
- psp_table_rights
- psp_view_rights
- psp_users
PSQL Security

You may access data from more than one database provided the databases are on the same machine. However, you can be logged in to only one database at a time. The following situations apply to database access based on the security of each database.

<table>
<thead>
<tr>
<th>Security for Logged-in Database</th>
<th>Security for Database Not Logged In</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Off</td>
<td>Access granted for all rights.</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>User name and password must be identical in both databases.</td>
</tr>
<tr>
<td>Off</td>
<td>On</td>
<td>Access denied.</td>
</tr>
</tbody>
</table>

The security discussion in the rest of this chapter applies only to the MicroKernel Engine. If you are interested only in the Relational Engine, you can skip to Data Encryption.

Available Models for the MicroKernel Engine

The authentication and authorization models that are available for MicroKernel Engine include the following:

- **Classic**
- **Mixed**
- **Database**

This topic, and the security discussion in the rest of this chapter, applies to applications that use only the MicroKernel Engine. The terms credentials, login credentials, or user credentials refer to a valid user name and password pair.

PSQL v12 supports OS-independent database authentication and authorization capabilities for the MicroKernel Engine. The original (operating system) authentication model is still available in this release, but now you can instead choose a model in which Btrieve users and privileges are not derived from file system users and privileges. You can allow users access to the database without allowing them operating system access to the data files.

Current Btrieve applications can take advantage of the new security models without requiring any changes to the application code.

Classic

Classic security is the Btrieve security model that was provided in previous releases of the product. For Btrieve users, authentication is performed by the operating system, and data access privileges are determined by file system rights for the given user. The only authorization capability provided by the database engine independent of the operating system is Btrieve owner names, which are essentially file access passwords.

Under this security model, any user who is authenticated by the operating system has the same rights to access the data through Btrieve as he or she has to read and/or write the data files through the operating system. Btrieve owner names are an exception to this rule, allowing an additional level of authorization. However, this level of authorization is not related to the user’s identity. It is related only to whether the application or the user can supply the owner name for a given file.

For more information on Btrieve owner names, see Owner Names.
Setting up Classic Security

Under Classic security, you set up database users and access permissions simply by creating users and assigning file permissions in the operating system. There are no separate actions to take to configure the database engine.

Refer to your operating system documentation for instructions on how to set up user accounts and assign permissions.

Mixed

In the Mixed security model, when a database login occurs, the database engine passes the user name and password entered by the user to the operating system authentication service. If the operating system authenticates the user name and password, then the database engine uses the users and rights table for the database to determine the specific access rights of the given user. Thus, each user's data access privileges must be defined within the database. In turn, the database engine enforces the defined privileges regardless of the given user's file system privileges on the data files.

Database authorization for Btrieve applications is provided by extending the Relational Engine security model so that it also can be used for Btrieve applications. The ability to create and define users and set permissions is provided through PCC and through SQL statements such as GRANT, REVOKE, ALTER GROUP, CREATE USER, ALTER USER, DROP USER.

Under the mixed security model, any user names defined in the database must correspond exactly with the same user names defined in the operating system. During a database login, the database engine simply passes the user name and password entered by the user to the operating system authentication module. If the operating system authenticates the credentials, then the database uses its own users and rights table to determine the specific access permissions of the given user. Each user must be added to the database. Instead of defining individual permissions for each user, you can define the permissions once using the default group PUBLIC. Valid users automatically inherit the permissions granted to the PUBLIC group.

For detailed procedures on how to set up a Mixed security environment, see Setting up Mixed or Database Security.

Database

Under the Database security model, the database engine authenticates and authorizes users of Btrieve data files. A user's ability to connect to and access data is unrelated to the user's operating system account identification and file system privileges, as long as the user can successfully log in to the computer on which his/her application runs.

Database authentication and authorization for Btrieve applications is provided by extending the Relational Engine security model so that it also can be used for Btrieve applications. The ability to define users and set permissions is provided through PCC and through SQL statements such as GRANT and REVOKE.

Note To create new databases, a user is still required to have administrator-level privileges in the operating system.

For detailed steps on how to set up a Database security environment, see Setting up Mixed or Database Security.
Notes on the Mixed and Database Security Models

For each database, a set of users must be defined, and for each user, a set of access permissions must be defined. The simplest case for assigning permissions is to assign them to the special group PUBLIC. All users inherit the default privileges from PUBLIC. In addition, you must specify the file system directory or directories that contain the data files that should be considered as members of the given database.

The database engine (or operating system in the case of Mixed security) performs user authentication and authorization for each attempt to access any data file within the directory tree. Without this association between databases and directories, when a Btrieve application attempts to open a specific data file, the database engine has no database context from which to determine the applicable set of defined users and permissions.

You can use the Mixed or Database security models only with Btrieve data files that reside in directories that have been defined as belonging to a given database, including the default database DefaultDB described in The Default Database and the Current Database. Data files residing in directories that have not been associated with a database can be accessed only with the Classic security model.

One of the primary advantages of these models is the ability to restrict operating system users’ access to the data files, while still allowing full access to the data through the database engine. In contrast, according to the Classic model, any user permitted to add records to the data file must necessarily also have the ability to copy or delete the data file from the operating system.

Note: Since the Workgroup engine performs no operating system authentication, the behavior of the Classic and Mixed security policies using the Workgroup engine are the same. If you wish to secure a Btrieve database using the Workgroup engine, you must use the Database security policy.

Setting up Mixed or Database Security

Migrating to mixed or database security requires that you make a number of choices and plan carefully. In a well-established environment, you may have to plan how your Btrieve files will be grouped together into databases, and schedule the migration so that you do not disrupt your production environment.

For complete information on making a transition from Classic to Mixed or Database security, see Security Tasks. The next section provides a brief overview of the material contained in the Security Tasks.

Owner Names

An owner name is a password required to gain access to a Btrieve file. There is no relation between an owner name and any system user name or relational database user name. You should think of an owner name as a file password.

A “short” owner name can be up to 8 bytes. A “long” owner name can be up to 24 bytes. Note, however, that once a long owner name is specified, the data file cannot be read by a database engine prior to PSQL v10.10. Also, a data file with a long owner name cannot be rebuilt to a file format prior to 9.5 unless the owner name is first removed. An owner name, long or short, with less than the maximum allowed bytes is padded with spaces to the maximum length (8 or 24 bytes).

PCC currently does not provide a way to specify an owner name through the security properties of a file. However, you can use a GRANT statement in PCC SQL Editor to supply an owner name. See Owner Name in SQL Engine Reference. You can also set or clear an owner name on a file with the Maintenance Utility or the Function Executor utility. See Manipulating Btrieve Data Files with Maintenance.
An owner name can have several attributes, as shown in Table 23.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read-only</td>
<td>Without specifying the password, users can perform data access operations that do not modify the data file.</td>
</tr>
<tr>
<td>Read-only encrypted</td>
<td>Without specifying the password, users can perform data access operations that do not modify the data file. When you set this option, the database engine encrypts every record in the file using the owner name as a key. Records added later are also encrypted.</td>
</tr>
<tr>
<td>Normal</td>
<td>Without specifying the password, users cannot perform any file access operations.</td>
</tr>
<tr>
<td>Normal encrypted</td>
<td>Without specifying the password, users cannot perform any file access operations. Any records inserted or updated are encrypted using the password. When you set this option, the database engine encrypts every record in the file using the owner name as a key. Records added later are also encrypted.</td>
</tr>
</tbody>
</table>

**Remarks**

When you first set an owner name with encryption on a file, the database engine encrypts the entire file. The larger the file is, the longer this procedure takes.

Data access operations to an encrypted file are slower than to a normal file. The database engine must decrypt each page as it reads it from disk, and encrypt it again before writing it back to disk.

---

**Caution** Remember and keep track of a file's owner name, especially with encryption turned on. There is no way to find out the owner name, and no way to access the data without it.

**Owner Names and Security**

If you have a Btrieve owner name set on a file that is a table in a secure database, the Master user of the database must use the owner name in any GRANT statement to grant privileges on the given table to any user, including the Master user.

After the GRANT statement containing the owner name has been issued for a given user, that user can access the specified table by logging into the database, without specifying the owner name each time.

If a user tries to access a table through ODBC that has a Btrieve owner name, the access will not be allowed unless the Master user has granted privileges on the table to the user, with the correct owner name in the GRANT statement.

If a table has an owner name with the Read-Only attribute, the Master user automatically has SELECT rights on this table without specifically granting himself/herself the SELECT rights with the owner name.

If no owner name is set on a data file, when relational security is enabled on that file, Btrieve access to the file is no longer permitted. You must set an owner name on that file in order to restore Btrieve access for those users who can supply the owner name when accessing the file. This behavior prevents default Btrieve users from circumventing relational security.
Examples
For examples of granting access to files with Btrieve owner names, see GRANT in SQL Engine Reference.

Accessing Data in More Than One Database
You may access data from more than one database provided the databases are on the same machine. However, you can be logged in to only one database at a time. The following situations apply to database access based on the security of each database.

Table 24  Access Rights to Databases Based on Security Settings

<table>
<thead>
<tr>
<th>Security for Logged-in Database</th>
<th>Security for Database NotLogged In</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled (either mixed or database security model)</td>
<td>None</td>
<td>Access granted for all rights.</td>
</tr>
<tr>
<td>Enabled (either mixed or database security model)</td>
<td>Enabled (either mixed or database security model)</td>
<td>User name and password must be identical in both databases.</td>
</tr>
<tr>
<td>None</td>
<td>Enabled (either mixed or database security model)</td>
<td>Access denied.</td>
</tr>
</tbody>
</table>
Planning Security for the MicroKernel Engine

After you install PSQL v12, the default behavior for security is the same as the previous release. That is, the database engine uses Classic or OS-based authentication and authorization. Any user with permission to access a given data file through the operating system will have the same level of permission to access the data records contained within the file, unless you are using Btrieve owner names to restrict access to the data files.

This section describes the steps you must follow to set up the default database, authorized users, and other aspects of the Btrieve security policies.

Available Options

There are three security options available to you. The features of these options are described next to help you choose which is best for you. Encryption is optional in every configuration.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Classic</th>
<th>Mixed</th>
<th>Database</th>
</tr>
</thead>
<tbody>
<tr>
<td>Administrator must set up separate operating system (OS) and database user accounts for each user</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Database user accounts are derived directly from OS user accounts</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Users’ data access rights are unrelated to users’ file system rights; administrator must assign data access privileges through the database to each user</td>
<td></td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Users’ data access rights are derived directly from OS users’ file system rights</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Supports automatic login dialog for entering database user name and password from any Windows application based on PSQL</td>
<td>✓¹</td>
<td>✓¹</td>
<td>✓</td>
</tr>
<tr>
<td>Database accepts successful OS login as valid database user</td>
<td>✓</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>User must log into database separately from logging into computer</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

¹ The login dialog may appear if the requester cannot establish an identity through the operating system.

Under **Database** security, database user accounts are completely unrelated to OS user accounts.

In contrast, under **Classic** security, a user who successfully logs into the computer has access to the database contents, at whatever level of file system rights that the user has been assigned to the file that contains the data.

Lastly, the **Mixed** security policy has aspects of both of the other policies. Under this scheme, users log in using their OS user names and passwords, but then the users access rights to the data are governed by user permissions set up in the database.

Choosing Your Policy

This section describes some of the major reasons you might choose one security policy over another.
Reasons to Choose Classic

- You are comfortable with your users having file system access to your data files. For example, any user with rights to delete records from the data file can also delete the entire file from your operating system.
- You want the minimum administrative hassle; you don't want to set up both OS user accounts for each user and at least one database account.
- You do not need to have a variety of data access rights that vary from each user's file system rights.
- You don't want your users to have a separate login for the database.

Reasons to Choose Mixed

- You don't want your users to have a separate login for the database.
- You want to prevent valid database users from having any rights to the data files on the operating system. For example, you can prevent users who have all rights in the database from having rights to delete data files from the operating system.
- You are willing to set up database user accounts that have the same user names as OS user accounts, and you are willing to assign permissions to each database user. If you choose, all of your users can have the same level of permissions by inheriting them from the special group PUBLIC.

Reasons to Choose Database

- You want to have a separate login for the database. That is, after logging into the operating system, users must log in again to the database. This behavior is useful when some authorized computer users are permitted access to the database and some are not.
- You want to prevent valid database users from having any rights to the data files on the operating system. For example, you can prevent users who have all rights in the database from having rights to delete data files from the operating system. You can also achieve this goal using the Mixed security policy.
- You want database user accounts that use different names than the operating system accounts. For example, operating system user “jsmith” might be required to log in to the database as “john.”
- The users and their permissions stay with the database, not with the server or machine. This allows you to move a database from one machine to another without having to re-create the users and their permissions for the database.

Preparing to Set Up Security

Setting up security for the MicroKernel Engine is a simple process, but it affords enough flexibility that some preparation is necessary. This section describes the information you should know before you begin to set up Btrieve security.

How Many Databases?

For Mixed or Database security, you must either assign all users the same level of permissions, or create a set of defined users for each database.

In some cases where your Btrieve data files encompass two or more completely unrelated bodies of data, you may want to set up two or more separate databases, each with its own set of authorized users. Generally speaking, however, you want to minimize the number of separate databases so that you do not have to create and maintain multiple sets of defined users. Often, a single database is sufficient.
permissions within the database will allow you to regulate each user’s access to the database, so you do not need to create separate databases just to limit certain users’ access.

If you determine that you need only one database, you may use the pre-existing database, **DefaultDB**, as the database associated with your Btrieve files. You may also set up your own named database instead.

**Where are the Data Files?**

You associate a Btrieve data file with a database by specifying the directory containing the data file as a **Data Directory** for the given named database. Thus, you need to know the directories containing all the data files that you want to associate with the database. If all the data files reside in a sub-directory tree within a specific directory, all you need to know is the top-level directory path name. You can even use "C:" if you wish to include all data files on the hard drive.

**What are the User Names?**

If you plan to use **Mixed** security, you must either assign all users the same permissions, or set up user accounts for the users whose rights differ. If you are going to set up individual users, you must have a list of the operating system user names that you want to make into database user names. The database user names that you set up must match the operating system user names exactly. You can always add additional user names later, but it is more efficient to create several users at once.

**What Security Policy?**

Before you set up security, you must know what policy you plan to use. The setup process varies somewhat for each policy. Considerations in choosing a policy are presented in **Choosing Your Policy**.

**Process Overview**

This section outlines the high-level procedure used to set up security for a database. Detailed, step-by-step instructions are provided in the section that follows.

1. **Preparation.** As specified above in **Preparing to Set Up Security**, gather the information you need and make the decisions necessary to get started. How many databases? Where are the Btrieve files located? What are the user names? What security policy will you use?

2. **Select a database to use with your Btrieve files, and populate the database with the data directory specifying the location of your data files. This step is only necessary for Mixed or Database security.**

   For details on this step, see **To use an existing database, including the pre-defined DefaultDB, with your PSQL files** in **PSQL User's Guide**.

3. **Turn on security.**

   For details on this step, see **To turn on security using PSQL Explorer** in **PSQL User's Guide**.

4. **Create users and permissions.** Using SQL statements or PCC, create your user accounts and/or relevant user privileges. This step is only necessary for Mixed or Database security.

   For the fastest, easiest way to grant users access, see **To assign permissions to all users using PSQL Explorer** in **PSQL User's Guide**.

5. **Set the Btrieve Security for your database to Mixed or Database.**

   For details on this step, see **To set or change the security policy for a database** in **PSQL User's Guide**.
Secure the data files in the operating system. For Mixed or Database security, users now can access the data without having any rights to access the data files in the operating system. Refer to your operating system documentation for information on securing access to files.

Summary of Tasks for MicroKernel Engine Security

The following table illustrates the basic level of effort required using the different security models. The tasks required to implement the security models, see Security Tasks.

Table 26 Summary of Security Set-up Tasks

<table>
<thead>
<tr>
<th>Security Model</th>
<th>Authentication/Authorization</th>
<th>Summary of Behavior and High-Level Setup Tasks</th>
</tr>
</thead>
</table>
| Classic        | Operating system/Operating system | - Give users file permission access to all database files.  
|                |                              | - Add an owner name to Btrieve files to further limit access (optional) |
| Mixed          | Operating system/Database    | - Note that this security model behaves the same as Classic when using the Workgroup engine.  
|                |                              | - Set up users in operating system. Users will be authenticated against this user name and password.  
|                |                              | - If you want individual user security, set up like-named users in the database security using the PSQL Control Center. Although authentication occurred at OS level, database permissions are stored in the database, so the operating system user name and database user name must match.  
|                |                              | - Define each user’s database permissions using PSQL Control Center or SQL statements. Alternatively, define a set of rights for the group PUBLIC. Each authenticated OS user inheriting from the group PUBLIC will have the same rights as PUBLIC. No user can have rights defined that are lower than that of PUBLIC. |
| Database       | Database/Database             | - Operating system user names and passwords are not relevant to the PSQL database security.  
|                |                              | - Set up users using the PSQL Control Center utility or SQL statements.  
|                |                              | - Define the database permissions using PSQL Control Center or SQL statements.  
|                |                              | - Using the PSQL Control Center Configuration tool, specify how authentication credentials are passed. This step refers to the new configuration parameters Prompt for Client Credentials and Allow Client-stored Credentials. |
MicroKernel Engine Security Quick Start

This section provides step-by-step instructions on the fastest, easiest way to secure your Btrieve data files in the operating system while still allowing database users to access the data.

When this procedure is complete, you can revoke operating system user rights to the data files without affecting database user rights to access the data through an application.

Note You must be logged into the computer where the database engine is installed, as an operating system user with administrative rights or as a user who is a member of the Pervasive_Admin security group.

1 Start PSQL Control Center (PCC). For how to start PCC, see Starting PCC on Windows in PSQL User's Guide.

2 If the database engine you wish to work with is not registered with PCC, register it now. For how to register a database engine, see To register a remote server engine in PSQL User's Guide.

3 Expand the databases for the registered engine (click the expand icon to the left of the node).

4 In PCC, right-click the database DefaultDB, then click Properties.

5 Click Directories then click New.

6 Type a path for the Btrieve files then click Apply.

   If your files are spread over many directories, specify a high-level directory that they all have in common. You can specify a root level if necessary, but doing so includes in DefaultDB all Btrieve files at the root level and its subordinate directories. For example, a root level could be C:\ for Windows. See To use an existing database, including the pre-defined DefaultDB, with your PSQL files in PSQL User's Guide.

   You do not need to enter every directory, just the lowest level directory that is common to all Btrieve files you want to include in the database.

7 Enable security on DefaultDB: click the “Security” node on the Properties dialog tree.

8 Click the “Database Security” tab.

9 Click Enable Security.

10 Type a password that you wish to use for the Master user, twice as prompted. Click OK.

   Now security is turned on, but access is based on OS user rights by default, so your users currently have the same access that they had before. The next step addresses this situation.

   Note that passwords are limited to a maximum of 8 bytes. You may use any displayable character in a password except for the semicolon (;) and the question mark (?).

11 Click OK to close the Properties dialog.

12 Expand the Groups for DefaultDB (click the expand icon to the left of the node), then right-click the group PUBLIC.

13 Click Properties then Permissions in the tree.

14 Click the “Database” tab.
Click the desired permissions.

For example, if you want to grant read-only rights to all authenticated users, click **Select**. This option will give all users read-only rights to the data. To give all users update permission, click **Update**, and so forth.

If you need to grant individual users varying rights, then you must create group accounts (if desired) and individual user accounts using the GRANT statement in SQL or using PCC. (See Security Tasks.

Click **OK**.

Right-click the database DefaultDB, then click **Properties**.

Click **Security** then click the Btrieve Security tab.

19 Click **Mixed** then **OK**.

---

**Note**

Do not change the Btrieve Security policy setting until you have completed step 15 as instructed. If you have not created user accounts or granted rights to the group PUBLIC, changing the security policy will prevent all your users from accessing the data.

---

You have now granted login access only to those users who are authenticated by the operating system, and you have specified that the access rights of those users are defined by the permissions you granted to them in the database.

---

20 Secure the data files in the operating system according to your operating system instructions. You can now deny operating system users from having any rights to the data files, without affecting their ability to access the data through the database engine.

---

**Caution**

Be sure to secure the data files in the operating system. If you do not perform this step, the users still can access the files through the operating system with the same level of permissions that they had prior to this procedure. You must revoke the users’ operating system privileges to the data files if you want to prevent users from being able to delete or modify the files directly.
Security Tasks

Data Encryption

PSQL v12 supports encrypting the database-related network traffic that occurs when using PSQL. This type of encryption is often called wire encryption because it protects the data when it is traveling on the network wire, or on any intervening network infrastructure, including wireless. While the use of wire encryption is not required, it provides additional deterrence against unauthorized access to the data transmitted by your application over a network.

This encryption feature is not directly related to the security models. Any of the security models can be used with or without wire encryption turned on.

Configuration Parameters for Wire Encryption

There are two configuration settings associated with wire encryption. The settings must be configured at each client machine as well as at the server. For more information on these settings, see

- Wire Encryption
- Wire Encryption Level

To access wire encryption settings

1 In PCC PSQL Explorer, perform one of the following:
   - For a server, right-click the server name under the Engines node. (Click the plus (+) signs to expand the nodes.)
   - For a client, right click on MicroKernel Router under the Local Client node. (Click the plus (+) signs to expand the nodes.)

2 Click Properties.

3 Click Access in the tree.

Encryption Notes

This release of the product uses a well-known and time-tested public domain encryption algorithm called "Blowfish" to perform the encryption before data passes over the network.

Encryption using a 40-bit key provides the least amount of protection for the data. Encryption using a 56-bit key is more difficult to compromise. Finally, encryption using the 128-bit key is the generally considered very difficult to compromise.

Note Using encryption slows the network throughput of your data.

Backward Compatibility

Because previous versions of PSQL did not support wire encryption, they will be unable to communicate with a client or server from this release that requires encryption. Any client or server that does not support encryption will return an error if it attempts to connect to a client or server that requires encryption.
Setting Up Encryption

Before turning on the encryption settings in your environment, think about your encryption needs first. You can set up your encryption environment in a variety of ways, depending on your situation. There are four general schemes possible:

- no encryption
- all communications encrypted
- encryption to/from specific clients
- encryption to/from specific servers

No Encryption

First of all, consider whether your data has characteristics that would favor encryption. Is your data confidential or proprietary? Is it valuable in the hands of unauthorized users? Can it be used to harm your organization? If you answer no to these questions and others like them, then your data may not need to be encrypted at all. Under these circumstances, there may be no reason to incur the performance trade-off that encryption entails. If you aren’t sure, consult a data security expert.

Assuming your data does need to be protected, you still may not need encryption. If your applications run solely on a LAN, and you are comfortable with the physical security of your network infrastructure, encryption may not be necessary.

Encryption to/from Specific Clients

Now suppose that you have one major customer at a remote site that has a connection to your database. You may wish to use encryption only for the communications that go to/from that remote client. You can achieve this affect by setting **Wire Encryption** at the remote client to **Always** and setting the server values accessed by that remote client to **If Needed**. All your internal clients would be set to **Never**. Thus, the servers will only use encryption when communicating with the remote client that requires encryption.

Encryption to/from Specific Servers

Now, suppose the situation is reversed and your environment includes one or more remote servers that are accessed by network infrastructure that you do not trust 100%. In this case, you can set those server values to **Always**, and set the local client values to **If Needed**. The result is encrypted communications only to those remote servers that require it.

All Communications Encrypted

Finally, if your PSQL applications often run over WAN, VPN, or other external networks that you do not trust 100%, then you may wish to encrypt 100% of your database communications. In this scenario, you would set **Wire Encryption** to **Always** at all clients and servers.

Choosing an Encryption Level

Once you have decided which clients and servers require encrypted communications, you must decide what level of deterrence is appropriate for your needs.

While Actian Corporation cannot offer advice regarding the encryption level that meets your specific needs, we can provide some guidelines to help inform your discussions with an appropriate data security expert.
PSQL Security

experts. These guidelines do not represent a guarantee or warranty from Actian Corporation that no third party will be able to intercept and/or decode your encrypted data. As with any encryption scheme, there is no such thing as an “unbreakable” code, only varying levels of difficulty to compromise different types of encryption. The 128-bit encryption used by PSQL would be considered “very difficult” to decode using techniques and equipment available to a highly sophisticated individual hacker.

**Low (40-bit) Encryption**

Consider using this level of encryption in cases where your data has limited ability to harm your organization or your customers if it falls into the wrong hands. Another reason to consider a Low level of encryption is if you wish simply to prevent a casual observer on your network from being able to read your data as it travels over the wires.

**Medium (56-bit) Encryption**

Consider using this level of encryption in situations where you believe you need somewhat more protection than against just a casual observer, but you do not believe you require the strongest level of security.

**High (128-bit) Encryption**

Consider using this level of encryption in situations where your data contains very sensitive information such as credit card numbers, social security numbers, financial account numbers, or other information protected by law. Especially consider this level of encryption if your database is associated with an entity on the network that is well-known to contain sensitive data, such as an Internet shopping web site or an online securities brokerage web site. Consider this level of encryption if your organization has previously suffered attempts to compromise its data security.

**Effects of Encryption**

Using encryption reduces client/server performance. With encryption turned on, each piece of data must be encoded at the source and decoded at the destination. This process requires additional CPU cycles when compared to the same operations performed without encryption. The level of encryption should not affect the performance. The performance drop in using encryption is roughly the same no matter which of the three encryption levels you choose.

**Owner Name Encryption**

PSQL offers encryption of data files on disk. To require that your data files be encrypted when written to disk, you must set an owner name on each file.

See Owner Names for more information.
Understanding Logs, Backups, and Data Restoration

PSQL provides several powerful features to ensure data integrity and to support online backups and disaster recovery.

- Transaction Logging and Durability
- Understanding Archival Logging and Continuous Operations
- Using Archival Logging
- Using Continuous Operations
- Data Backup with Backup Agent and VSS Writer
Transaction Logging and Durability

PSQL offers two levels of data integrity assurance for database operations that involve transactions: Transaction Logging and Transaction Durability.

This section contains the following sub-topics:

- Using These Features
- Feature Comparison
- Which Feature Should I Use?
- How Logging Works
- See Also

Using These Features

Both of these features can be turned on or off in the database engine using configuration within PSQL Control Center, or programmatically using the Distributed Tuning Interface. See Transaction Durability and Transaction Logging.

The default value for Transaction Durability is Off, and the default value for Transaction Logging is On.

Feature Comparison

Both features offer multifile transaction atomicity, to ensure that the data files remain consistent as a set and that incomplete transactions are never written to any data files.

Atomicity means that, if any given data operation within a transaction cannot successfully complete, then none of the operations within the transaction are allowed to complete. An atomic change does not leave partial or ambiguous effects in the database. Changes to individual files are always atomic whether Transaction Logging and Transaction Durability are on or off. But transactions make it possible to group changes to multiple files into one atomic group. The atomicity of these multifile transactions are assured by the MicroKernel only when using transactions in your application, and Transaction Logging or Transaction Durability is turned on.

In addition to these benefits, Transaction Durability guarantees that, in the event of a system crash, the data files will contain the full results of any transaction that returned a successful completion status code to the application prior to the crash.

In the interest of higher performance, Transaction Logging does not offer this guarantee. Whereas Transaction Durability ensures that a completed transaction is fully written to the transaction log before the engine returns a successful status code, Transaction Logging returns a successful status code as soon as the logger thread has been signaled to flush the log buffer to disk.

Transaction Logging is a sub-set of Transaction Durability; that is, if Transaction Durability is turned on, then logging takes place and the Transaction Logging setting is ignored by the database engine.
The main differences between Transaction Logging and Transaction Durability are shown in the following tables:

Table 27  Transaction Logging vs. Transaction Durability: Benefits

<table>
<thead>
<tr>
<th>Feature</th>
<th>Guaranteed data consistency and transaction atomicity across multiple files</th>
<th>Guaranteed commit for all completed transactions that have returned a successful status code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Logging</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Transaction Durability</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 28  Transaction Logging vs. Transaction Durability: Function

<table>
<thead>
<tr>
<th>Feature</th>
<th>Timing of log buffer writes to disk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Logging</td>
<td>The log buffer is written to the log file when the log buffer is full or Initiation Time Limit is reached. A successful status code for each End Transaction operation is returned to the application as soon as the logger thread has been signaled to flush the buffer to disk.</td>
</tr>
<tr>
<td>Transaction Durability</td>
<td>The log buffer is written to the transaction log file with each End Transaction operation. A successful status code for each End Transaction operation is not returned to application until the log disk write is successful. For insert or update operations that are not part of a transaction, the log buffer is written to the log file when the log buffer is full or Initiation Time Limit is reached.</td>
</tr>
</tbody>
</table>

**Which Feature Should I Use?**

For the fastest performance, you want to use the lowest level of logging that meets your transaction safety needs. The best way to determine your appropriate level of logging is to ask your application vendor. If you have multiple applications that use PSQL on the same computer, you must use the highest level of logging required by any of the applications.

If you only have one data file, or if none of your applications perform transactions involving multiple data files, you generally do not need to use Transaction Durability or Transaction Logging. Under these circumstances, PSQL guarantees the internal consistency of each data file, with or without logging.

**Transaction Logging**

Turn on Transaction Logging if at least one of your PSQL applications performs transactions across multiple data files. Without Transaction Logging, PSQL cannot guarantee multifile atomicity of transactions or multifile data integrity.

In the event of a system crash, this level of logging does not guarantee that every completed transaction has been written to the data files.

**Transaction Durability**

Turn on Transaction Durability if at least one of your PSQL applications requires that completed transactions across multiple data files be absolutely guaranteed to have been written to the data files under almost any circumstances.

In the event of a system crash, this level of logging guarantees that every transaction that has been successfully completed has been written to the data files.
**How Logging Works**

Note that these features ensure atomicity of transactions, not of operations. If you are using SQL, a transaction is defined as a set of operations that take place between a \textit{BEGIN} statement or \textit{START TRANSACTION} statement, and an \textit{END} or \textit{COMMIT} statement. If you are using Btrieve, a transaction is defined as a set of operations that take place between a Start Transaction operation and an End Transaction operation.

All data file inserts and updates are stored in the log buffer. When a transaction is completed (Transaction Durability) or when the buffer gets full or the \textit{Initiation Time Limit} is reached (Transaction Durability or Transaction Logging), the buffer is flushed to the transaction log file.

In the case of Transaction Logging, when the engine receives the operation ending the transaction and successfully signals the logger thread to flush the log buffer to disk, the engine returns a successful status code to the application that initiated the transaction. In the case of Transaction Durability, the engine does not return the successful status code until the logger thread signals that it has successfully written the buffer to disk.

Transaction log file segments are stored in the location specified in the setting \textit{Transaction Log Directory}. The log segments are named *.LOG, where the prefix can be 00000001 through FFFFFFFF.

**Note** All operations, regardless of whether they take place within a transaction, are written to the log file when Transaction Logging or Transaction Durability is in effect. However, only operations executed within a transaction are guaranteed to be atomic. In the case where a system crash has occurred and the transaction log is being rolled forward, only completed transactions are committed to the data files. All operations without an associated End Transaction operation are rejected, and are not committed to the data files.

**Tip** If your database is highly used, consider configuring your system to maintain the transaction logs on a separate physical volume from the volume where the data files are located. Under heavy load, performance is typically better when the writes to the log files and to the data file are split across different drives instead of competing for I/O bandwidth on a single drive. The overall disk I/O is not reduced, but the load is better distributed among the disk controllers.

You can specify the location of the transaction logs using the configuration setting \textit{Transaction Log Directory}.

If a system failure occurs after the log file has been written but before the “committed” operations are flushed to the data files in a system transaction, the “committed” operations are not lost. In order to flush the committed operations, the affected files need to be opened and operations performed after the system failure. When the files are opened and operations attempted, it is then that the data is rolled forward to the files affected at the time of system failure. Simply restarting the database engine will not invoke the roll forward operation nor will it make the data consistent.

**Note** Log files associated with the rolled forward files will not be automatically deleted, as they may be associated with more than one data file.
This feature allows individual client transactions to receive a successful status code as soon as possible while at the same time taking advantage of performance gains offered by grouping multiple client transactions together and writing them to the data files sequentially.

If your database server suffers a disk crash of the volume where the data files are stored, and you have to restore the data from an archival log, the engine does not roll forward the transaction log file. The archival log contains all the operations in the transaction log, so there is no need to roll forward the transaction log.

Tip After a system failure, open all data files and perform a stat or read operation on those files. Once you are certain that all data has been restored, old log files may then be stored in a safe location.

See Also

For further information, see:

Transaction Durability
Transaction Logging
Transaction Log Directory
Understanding Archival Logging and Continuous Operations

The product offers two mutually exclusive features to support online backups and disaster recovery.

<table>
<thead>
<tr>
<th>If your situation is like this...</th>
<th>... use this feature:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You must keep your database applications running while performing backups.</td>
<td>Continuous operations</td>
</tr>
<tr>
<td>You are able to shut down the database engine to perform backups</td>
<td>Archival logging</td>
</tr>
</tbody>
</table>

Archival Logging allows you to keep a log of database operations since your last backup. In case of a system failure, you can restore the data files from backup then roll forward the changes from the log file to return the system to the state it was in prior to the system failure.

**Caution** Archival logging does not guarantee that all your data files will be in a consistent state after restoring from an archival log. In the interest of speed, the database engine does not wait for a successful status code from the logging function before emptying the log buffer. Thus, in rare circumstances such as a full disk or a write error in the operating system, updates that were successful in the data files may not be recorded in the archival log. In addition, archival logging does not require you to log all of your files, so a transaction that updates more than one file may not be completely recorded in the archival log if you are only archival logging some of those files. As a result, one file may not be consistent with another. If you use transactions and require multifile transaction atomicity, see Transaction Logging and Durability.

Continuous Operations allows you to backup database files while the database engine is running and users are connected. After starting Continuous Operations, the database engine closes the active data files and stores all changes in temporary data files (called delta files). While Continuous Operations are in effect, you perform a backup of the data files. The delta files record any changes made to the data files while the backup is taking place.

When the backup is complete, you turn off Continuous Operations. The database engine then reads the delta file and applies all the changes to the original data files. The temporary delta file may surpass the size of the original data file if users make extensive changes to the file during continuous operation.

A file put into continuous operations locks the data file from deletion through the Relational Engine and the Microkernel Engine. In addition, the file is locked from any attempts to change the file structure, such as modifying keys and so forth.

**Note** Archival Logging and Continuous Operations are mutually exclusive features and cannot be used at the same time.

**Difference Between Archival Logging and Transaction Logging**

Transaction Logging is another feature designed to protect the integrity of your data in the event of a system failure, but it is not directly related to Archival Logging. You can have Transaction Logging in effect at the same time as either Archival Logging or Continuous Operations. Transaction Logging uses a short-term log file to ensure that transactions are safely written to disk. The transaction log is reset frequently as completed client transactions are rolled into the physical data files by way of system
transactions. In the event of a system failure, when the database engine starts up again, it reads the transaction log and flushes to the data files the transactions that were completed prior to the system failure.

The archival log is written to at the conclusion of each system transaction, so the archival log and the transaction log should remain properly synchronized unless a system failure occurs exactly during the system transaction.

For more information on Transaction Logging, see Transaction Logging and Durability.

**What if a File Restore is Needed**

In the event of a system crash that requires restoring data files from backup, Archival Logging allows you to restore from backup and then recover database activity up to the moment of the crash.

If you experience a similar crash without Archival Logging (for example if you use Continuous Operations to perform backups), then you will not be able to recover database activity that took place between the last backup and the system crash.

<table>
<thead>
<tr>
<th>If Archival Logging is...</th>
<th>... this much data will be unrecoverable after a crash:</th>
</tr>
</thead>
<tbody>
<tr>
<td>On</td>
<td>Unfinished transactions at the moment of failure.</td>
</tr>
<tr>
<td>Off</td>
<td>All database operations that have occurred after the last backup of the data files.</td>
</tr>
</tbody>
</table>

The remainder of this chapter describes the options and procedures associated with Archival Logging and Continuous Operations.
Using Archival Logging

This section explains the procedures you must follow to set up Archival Logging, make backups, and restore data files. It is divided into the following sub-topics:

- General Procedures
- Setting up Archival Logging
- Roll Forward Command

General Procedures

For Archival Logging to work properly, you must follow a clearly defined procedure to set it up, and another procedure in the event that a restore from backup is necessary.

Caution If any steps of the procedures are omitted or compromised, you may not be able to restore your data to its pre-crash state.

➢ To use Archival Logging properly

1. Turn on Archival Logging, if it is not already in effect. See Setting up Archival Logging for the detailed set-up procedure.

2. Shut down the database engine.

3. Backup the data files.

4. After a successful backup, delete all existing archival logs.

Caution Delete the corresponding log files before you resume working with the data files. Synchronizing the backup data files and the corresponding log files is a critical factor of successful recovery.

5. Restart the database engine.

➢ To restore data files from backup and apply changes from the archival logs

Note You cannot use this procedure to roll forward the archival logs if you experienced a hard disk crash and your archival logs and data files were both located on the lost hard disk.

1. When the computer re-starts after the system failure, ensure that the database engine is not running, and ensure no other database engine is accessing the data files you wish to restore.

2. Restore the data files from backup.

3. Start the database engine, ensuring that no applications of any kind are connected to the engine.
Using Archival Logging

Caution It is crucial that no database access occurs before the archival logs have been applied to the data files. Make sure no other database engine accesses the files. You must roll forward the archival logs using the same engine that encountered the system failure.

4 Issue the Roll Forward command as described in Roll Forward Command.

5 After the Roll Forward completes successfully, stop the database engine and make a new backup of the data files.

6 After you have successfully backed up the data files, delete the archival log files. You may now re-start the database engine and allow applications to access the data files.

Setting up Archival Logging

Setting up Archival Logging requires two steps:

- turning on the Archival Logging feature
- specifying the files to archive and their respective log files

Note To perform these procedures, you must have full administrative permissions on the machine where the database engine is running or be a member of the Pervasive_Admin group on the machine where the database engine is running.

➢ To turn on Archival Logging

1 Access Control Center from the operating system Start menu or Apps screen.

2 In PSQL Explorer, expand the Engines node in the tree (click the expand icon to the left of the node).

3 Right-click the database engine for which you want to specify archival logging.

4 Click Properties.

5 Click Data Integrity in the tree to display the settings for that category of options.

6 Click Archival Logging Selected Files.

7 Click OK.

A message informs you that the engines must be restarted for the setting to take effect.

8 Click Yes to restart the engine.

➢ To specify files to archive

You specify the files for which you want the MicroKernel to perform Archival Logging by adding entries to an archival log configuration file you create on the volume that contains the files. To set up the configuration file, follow these steps:

1 Create the directory \BLOG in a real root directory of the physical drive that contains data files you want to log. (That is, do not use a mapped root directory.) If your files are on multiple volumes, create a \BLOG directory on each volume.

For example, if you have data files located on C: and D:, and both drives are physical drives located on the same computer as the database engine, then you would create two BLOG directories, as next:
Logging, Backup, and Restore

C:\BLOG\nD:\BLOG\n
Note On Linux, the log directory must be named blog and must be created in the directory specified by the PVSW_ROOT environment variable (by default, /usr/local/psql).

2 In each \BLOG directory, create an empty BLOG.CFG file. You can use any text editor, such as Notepad, to create the BLOG.CFG file. On Linux, the file must be named blog.cfg (lowercase).

3 In each BLOG.CFG file, create entries for the data files on that drive for which you want to perform Archival Logging. Use the following format to create each entry:

```bash
\path1\dataFile1 [\=\path2\logFile1]
```

path1 The path to the data file to be logged. The path cannot include a drive letter.
dataFile1 The name of the data file to be logged.
path2 The path to the log file. Because the log file and the data file can be on different drives, the path can include a drive letter.
logFile1 The name of the log file. If you do not specify a name, the default value is the same directory and file name prefix as the data file, but replace the file name suffix with "log." You may specify a different physical drive, so that the log and the data files are not on the same drive. Each data file being logged requires a different log file.

A single entry cannot contain spaces and must fit completely on one line. Each line can contain up to 256 characters. If you have room, you can place multiple entries on the same line. Entries must be separated by white space.

Caution You must use a different log file for every data file that you wish to log. If you use the same log file for more than one data file, the MicroKernel cannot use that log file in the event that a roll-forward is needed.

If you do not provide a name for a log file, the MicroKernel assigns the original file name plus a LOG extension to the log file when you first open it. For example, for the file B.BTR, the MicroKernel assigns the name B.LOG to the log file.

Caution You are not required to log every file in your database. However, if your database has referential integrity (RI) rules defined, you must log all or none of the files involved in each RI relationship. If you log only a sub-set of the files involved in a given RI relationship, rolling the archival logs forward after a system crash may result in violations of your RI rules.

Examples

The following examples show three sample entries in the BLOG.CFG file on drive C. All three entries produce the same result: activity in the file C:\DATA\B.BTI is logged to the file C:\DATA\B.LOG.

```
data\b.bti
\data\b.bti=\data\b.log
\data\b.bti=c:\data\b.log
```
Using Archival Logging

The next example directs the engine to log activity in the file C:\DATA\B.BTI to the log file D:\DATA\B.LGF. This example shows that archival log files do not have to reside on the same drive as the data file and do not require the .LOG extension. (The .LOG extension is the default.)

\data\b.bti=d:\data\b.lgf

Tip Writing the log to a different physical drive on the same computer is recommended. If you experience a hard disk crash, having the log files on a different physical disk protects you from losing your log files and your data files at the same time.

The next example shows a BLOG.CFG file that makes the MicroKernel log multiple data files to a different drive (drive D:), assuming this BLOG.CFG file is on drive C:

\data\file1.mkd=d:\backup\  
\data\file2.mkd=d:\backup\file2.log  
\data\file3.mkd=d:\backup\file3.log

Roll Forward Command

The Btrieve Maintenance utility (GUI or BUTIL command line) provides a command allowing you to roll forward archival log files into the data files. See Performing Archival Logging.
Using Continuous Operations

Continuous Operations provides the ability to backup data files while database applications are running and users are connected. However, in the event of a hard drive failure, if you use Continuous Operations to make backups, you will lose all changes to your data since the last backup. You cannot use Archival Logging and the Maintenance utility Roll Forward command to restore changes to your data files that occurred after the last backup.

PSQL provides a backup command, BUTIL, for Continuous Operations. (Note that PSQL also provides a product, Backup Agent, to set and manage continuous operations. See the documentation provided with that product, Backup Agent Guide, for details.)

Note A file put into continuous operations locks the data file from deletion through the Relational Engine and the MicroKernel Engine. In addition, the file is locked from any attempts to change the file structure, such as modifying keys and so forth.

This section is divided into the following sub-topics:

- Starting and Ending Continuous Operations
- Backing Up a Database with BUTIL
- Restoring Data Files when Using Continuous Operations

Starting and Ending Continuous Operations

This section provides detailed information on the commands: Startbu and Endbu.

Table 30 Commands to Start and Stop Continuous Operation

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Startbu</td>
<td>Starts continuous operation on files defined for backup (BUTIL).</td>
</tr>
<tr>
<td>Endbu</td>
<td>Ends continuous operation on data files defined for backup. (BUTIL).</td>
</tr>
</tbody>
</table>

Caution The temporary delta files created by Continuous Operations mode have the same name as the corresponding data files but use the extension "^^^" instead. No two files can share the same file name and differ only in their file name extension if both files are in the same directory. For example, do not use a naming scheme such as INVOICE.HDR and INVOICE.DET for your data files. If you do, the MicroKernel returns a status code and no files are put into Continuous Operations.

Continuous operation mode does not significantly affect MicroKernel performance; however, using a server to back up files can affect performance.

To protect against data loss using Continuous Operation

1. Use the startbu command to put your files in continuous operation. See Startbu for an explanation of the command syntax with butil.
2. Back up your data files.
Using Continuous Operations

3 Use the `endbu` command to take your files out of continuous operation. See `Endbu` for an explanation of the command syntax with `butil`.

**Backing Up a Database with BUTIL**

This section provides detailed information on backing up a database using the following `butil` commands: `Startbu` and `Endbu`.

**Startbu**

The `butil startbu` command places a file or set of files into continuous operation for backup purposes.

**Format**

```
butil -startbu <sourceFile | @listFile> [/UID <name> /PWD <word>] [/DB <name>]```

- **sourceFile**: The fully qualified name of the data file (including the drive specification for Windows platforms) on which to begin continuous operation for backup. This fully qualified name must reside on the same machine as the one from which you are running `butil`. You cannot use mapped drives with the `startbu` command.

- **listFile**: The name of a text file containing the fully qualified names of files on which to begin continuous operation. Separate these file names with a carriage return/line feed. The file names may contain blank characters.

- **/UID <name> /UID <name>**: Specifies the name of the user authorized to access a database with security enabled.

- **/PWD <word> /PWD <word>**: Specifies the password for the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.

- **/DB <name> /DB <name>**: Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

---

**Note** The `startbu` command begins continuous operation only on the files you specify. You cannot use wildcard characters with the `startbu` command.

On Linux distributions, all “/” parameters use the hyphen ("-") instead of the slash. For example, the `/DB` parameter is `-DB`.

**File Considerations**

When selecting files for backup, we recommend that the temporary delta files created by Continuous Operations mode be excluded since they are open and in use during backup. If the delta files are included in the backup, they should be deleted before the database engine is started after the restore.

**Examples for Windows Server**

**Example A** The first example starts continuous operation on the COURSE.MKD file.
Logging, Backup, and Restore

For Windows Server:

butil -startbu file_path\PSQL\Demodata\course.mkd

(For default locations of PSQL files, see Where are the PSQL files installed? in Getting Started With PSQL.)

Example B The following example starts continuous operation on all files listed in the STARTLIST.FIL file.

butil -startbu @startlst.fil

The STARTLIST.FIL file might consist of the following entries:

file_path\PSQL\Demodata\course.mkd
file_path\PSQL\Demodata\tuition.mkd
file_path\PSQL\Demodata\dept.mkd

Endbu

The endbu command ends continuous operation on a data file or set of data files previously defined for backup. Issue this command after using the startbu command to begin continuous operation and after performing your backup.

Format

butil -endbu </A | sourceFile | @listFile | [/UID<name> | /PWD<word>> | [/DB<name>]]

/A If you specify /A, the utility stops continuous operation on all data files initialized by startbu and currently running in continuous operation mode.

sourceFile The fully qualified name of the data file (including the drive specification for Windows platforms) for which to end continuous operation.

This fully qualified name must reside on the same machine as the one from which you are running butil. You cannot use mapped drives with the endbu command.

@listFile The name of a text file containing a list of data files for which to end continuous operation. The text file must contain the fully qualified file name for each data file. Separate these file names with a carriage return/line feed. The file names may contain blank characters.

Typically, this list of data files is the same as the list used with the Startbu command.

/UID<name> Specifies the name of the user authorized to access a database with security enabled.

/PWD<word> Specifies the password for the user who is identified by uname. Pword must be supplied if uname is specified.

/DB<name> Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Note On Linux distributions, all “/” parameters use the hyphen (“-”) instead of the slash. For example, the /A parameter for is -A, as in butil -endbu -A.
**Example for Windows Server**

The following example ends continuous operation on the COURSE.MKD file.

```
butil -endbu file_path\PSQL\Demodata\course.mkd
```

However, you can also just enter `butil -endbu course.mkd` instead of the full path if your current directory is `f:\demodata`.

**Restoring Data Files when Using Continuous Operations**

If you are using Continuous Operations for your backup strategy, then you have no recovery log that can be used to recover changes since your last backup. All database changes since your last backup are lost, with the possible exception of any transactions stored in the transaction log. Any such transactions are automatically rolled forward by the database engine when it starts up.

➢ **To restore data and normal database operations**

1. Resolve the failure.
   Perform the maintenance required to make the failed computer operational again.

2. Restore the data files from backup, or restore the hard drive image from backup, as appropriate.

3. Re-install PSQL if it was not restored as part of a disk image.

   **Caution** If the delta files were included in the backup, they should be deleted before the database engine is started in the next step.

4. Re-start the database engine.

   Any database operations performed since the last backup must be performed over again.
Data Backup with Backup Agent and VSS Writer

In addition to the topics previously discussed in this chapter, both PSQL Server and PSQL Vx Server also provide the following solutions for data backup:

- **Backup Agent**
- **PSQL VSS Writer**

If your backup software is not aware of the Microsoft Volume Shadow Copy Service (VSS), you can use Backup Agent with your backup software. If your backup software is VSS aware, PSQL VSS Writer is automatically invoked during VSS backups. You do not need to use Backup Agent if your backup software is already VSS aware.

Backup Agent and PSQL VSS Writer can be used together, but there is no advantage in doing so. Your backup process will be more streamlined if you select one method or the other.

**Backup Agent**

Backup Agent is an optional product. By default, it is not installed. You must install it after you install PSQL Server.

Backup Agent provides a quick and simple method for you to set and manage Continuous Operations on your PSQL database files. Setting and managing Continuous Operations is a critical piece when backing up your PSQL databases without using Microsoft Volume Shadow Copy Service. Backup Agent handles setting and managing Continuous Operations on your open files so that your data is still available from your application during your backup. Once the backup procedure is complete, stopping Backup Agent automatically takes the files out of Continuous Operations and rolls in all the changes captured during the backup.

Backup Agent is compatible with many popular backup applications on the market. Note that the backup application must be able to issue commands to start and stop other applications (so that the commands can start and stop Backup Agent).

For details on Backup Agent, see Backup Agent Guide, which is available on the PSQL website.

**PSQL VSS Writer**

The Microsoft Volume Shadow Copy Service (VSS) consists of Writer, Provider, and Requestor components. PSQL v12 supports VSS with only a Writer component, PSQL VSS Writer.

PSQL VSS Writer is a feature of the database engine and is enabled for PSQL v12 Server. PSQL VSS Writer is available for use after that product is installed. PSQL VSS Writer is currently not available for use with PSQL Workgroup.

PSQL VSS Writer is available only on Windows operating systems. For more information on Volume Shadow Copy Service, refer to the Microsoft Website document, A Guide for SQL Server Backup Application Vendors.

**Overview**

During VSS snapshots, PSQL VSS Writer quiesces all disk I/O write activity to all PSQL data and transaction log files, regardless of the volume on which they reside. After the snapshot is taken, PSQL VSS Writer allows all disk I/O to resume; this includes any writes that were deferred during the quiesced period.
PSQL VSS Writer never quiesces disk I/O read activity, allowing normal database processing to continue during the quiesced period as long as writes are not required. PSQL VSS Writer operates normally during the backup phase, although performance may likely be reduced due to the backup activity of the VSS service and VSS Requestor.

The Microsoft Volume Shadow Copy facility allows Backup and Restore products to create a shadow copy for backup in which the files are in either one of the following states:

1. A well-defined and consistent state
2. A crash-consistent state (possibly not suitable for a clean restore).

Files in the VSS snapshot will be in the well-defined and consistent state if all of the following are true:

1. The file's writer is VSS-aware.
2. The Backup and Restore product recognizes and notifies the VSS-aware writer to prepare for a snapshot.
3. The VSS-aware writer successfully prepares for the snapshot.

Otherwise the writer's files are backed up in the crash-consistent state.

**VSS Writer Details**

The following items discuss specifics about PSQL VSS Writer.

- **Supported Operating Systems**
  
  The same Windows operating systems that support the PSQL server products also support PSQL VSS Writer. PSQL VSS Writer is functional on the same bitness as the machine's operating system and the installed PSQL server product. PSQL VSS Writer 32-bit is supported only on 32-bit machines, and 64-bit is supported only on 64-bit machines. If the bitness does not match, PSQL functions properly, but VSS Writer is unavailable.

- **Supported Backup Types**
  
  PSQL VSS Writer supports manual or automatic backups of data volumes. PSQL VSS Writer is supported on Full and Copy Volume backups. Incremental, Differential, and Log backups are not supported. VSS recognizes PSQL VSS Writer as a component. However, PSQL VSS Writer does not support component backups. If the VSS Requestor does call PSQL VSS Writer in a component backup, the VSS Writer performs the same actions as in a Full or Copy Volume backup.

- **Virtualized Environment Support**
  
  PSQL VSS Writer supports VSS Requesters that trigger VSS backups in virtualized environments. Performing a VM snapshot does not invoke a VSS backup.

- **Multiple Volume PSQL Data Files**

  PSQL files and transaction logs can reside on multiple volumes. When backing up PSQL files, remember to backup the transaction logs and related files on other volumes simultaneously. Files that are independent of one another may not need to be backed up at the same time as related PSQL files.

- **Backup Solution Compatibility**
To determine if a particular backup product recognizes PSQL VSS Writer and will notify the Writer to prepare for a snapshot, start a backup with the product. After the backup is in progress, consult the PVSW.LOG to determine if the PSQL VSS Writer logged the Frozen or Thawed states. If the backup and restore product did not notify PSQL VSS Writer to prepare for the backup, another solution must be used. For example, you could use Backup Agent to backup PSQL data files in the well-defined and consistent state.

- **PSQL VSS Writer and Restore Operations**
  
  Stop the PSQL services prior to performing a Restore operation with the Backup software. Failure to do so causes the VSS Writer to inform the VSS Requestor that it cannot participate in the Restore. Transaction logs will need to be restored along with the data files to guarantee the integrity of the data. If PSQL data and transaction log files are restored while PSQL is running, the results are unpredictable and could lead to data corruption.

- **PSQL VSS Writer and PSQL Continuous Operations**
  
  You may have an existing backup process that already uses PSQL Continuous Operations or Backup Agent. If you choose, you can continue to use that process with PSQL and PSQL VSS Writer. PSQL VSS Writer does not interfere with Continuous Operations or Backup Agent. However, there is no advantage to using both PSQL VSS Writer and Continuous Operations (or Backup Agent) together. Your backup process will be more streamlined if you select one method or the other.

  When PSQL VSS Writer is called and files are in Continuous Operations, be aware that VSS Writer operates independently from any Continuous Operations. If files are in Continuous Operations when a VSS backup is in progress, view PVSW.LOG after the backup completes. Ensure that the Frozen and Thawed states completed successfully and that the data is in a well-defined and consistent state.

  Also note that PSQL VSS Writer requires the Microsoft VSS framework. Backup Agent does not use the Microsoft VSS framework. Consequently, Backup Agent does not participate in the backup when the VSS framework calls PSQL VSS Writer and I/O operations are quiesced. Backup Agent must be added separately to the backup process. The backup process must also start and stop Backup Agent.
High Availability Support

Using PSQL in High Availability Environments

This chapter includes the following sections:

- Overview of Technologies
- Failover Clustering
- Migration
- Fault Tolerance
- Disaster Recovery
High Availability Support

Overview of Technologies

PSQL is compatible with numerous solutions that maximize uptime in physical and virtual environments. Such solutions continually evolve but can be classified generally as high availability, fault tolerance, and disaster recovery.

High Availability

The definition of “high availability” can differ depending on the software vendor that provides high availability solutions. In general, it refers to a systems design approach for a predictable baseline level of uptime, despite hardware failure, software failure, or required maintenance.

A common approach to ensure high availability in a physical environment is failover clustering. A common approach in a virtual machine (VM) environment is migration.

Failover Clustering

PSQL is designed to function as a resource in a failover cluster environment in which only one server node at a time accesses the shared storage subsystem. If the primary node fails, a failover (or switch) to a secondary node occurs. Failover clustering allows a system to remain available while you perform software upgrades or hardware maintenance.

PSQL is compatible with Microsoft Failover Cluster Services and with Linux Heartbeat. Refer to the documentation from those vendors for the specific manner in which they define and implement failover clustering. PSQL Server and PSQL Vx Server are the recommended editions for failover clustering.

See Failover Clustering.

Migration

In general terms, migration allows a running VM or application to be moved between different physical machines without disconnecting the client or application. The memory, storage, and network connectivity of the VM are typically migrated to the destination.

PSQL is compatible with the migration capability offered by Microsoft Hyper-V, VM ware vSphere, and Citrix XenServer. As long as host names remain the same after the VMs are moved, PSQL continues to operate normally. Refer to the documentation from those vendors for the specific manner in which they define and implement migration.

See Migration.

Fault Tolerance

While high availability aims for a predictable baseline level of uptime, fault tolerance is the uninterrupted operation of a system even after the failure of a component. Fault tolerance requires synchronized shared storage. In virtualized environments, the VM that fails must be on a different physical host from the VM that replaces it.

Fault tolerance can be achieved using just physical machines. However, virtual environments lend themselves so readily to maintaining virtual servers in lockstep with each other that exclusively physical environments are increasingly less common. PSQL Server is compatible with fault tolerance capabilities in an exclusively physical environment.
For virtual environments, PSQL is compatible with the fault tolerance capability offered by VMware vSphere and Citrix XenServer. Refer to the documentation from those vendors for the specific manner in which they define and implement fault tolerance.

See Fault Tolerance.

**Disaster Recovery**

Disaster recovery involves duplicating computer operations after a catastrophe occurs and typically includes routine off-site data backup as well as a procedure for activating vital information systems in a new location.

PSQL is compatible with major hypervisors that support disaster recovery technology that initializes backup physical or virtual machines. As long as all host names remain the same after the VMs are moved, PSQL continues to operate normally. This allows rapid server replacement and recovery time.

Refer to the documentation from the hypervisor vendors for the specific manner in which they define and implement disaster recovery.

See Disaster Recovery.

**Hardware Requirements**

For all of the technologies mentioned in this section, we recommend that you select servers, disk subsystems, and network components from the hardware compatibility list provided by the vendor. We follow this same practice when testing for compatibility with vendor products.
Failover Clustering

Failover clustering provides for multiple physical servers (nodes) to access a common, shared storage subsystem that contains one or more file shares or volumes. The failover services ensure that only one server controls the file shares or volumes at a time. Control of the shared storage subsystem is passed automatically from a failed server to the next surviving server in the cluster.

PSQL must be licensed separately on each cluster node where you install the database engine. This applies whether the node is a physical machine or a virtual machine. See also License Models in PSQL User’s Guide. (The failover clustering being discussed refers to Microsoft Failover Clustering or Cluster Service and to Linux Heartbeat. This is distinct from other high availability solutions that may incorporate VM-based clusters.)

This section contain the following topics:

- Microsoft Failover Clustering for Windows Server
- Linux Heartbeat
- Managing PSQL in a Cluster Environment

Microsoft Failover Clustering for Windows Server

This topic discusses adding the PSQL services to Failover Clustering and assumes the following:

- You know how to install and configure Failover Clustering and need only the information required to add and manage the PSQL services.
- You are familiar with using PSQL and its primary utilities such as PSQL Control Center (PCC).
- You can set up DSNs using ODBC Administrator.

Differences Between Windows Server Versions


Note the following differences with Windows Server 2012:

- In Failover Cluster Manager, “Services and Applications” are called “Roles.”
- Registry Replication is not supported for Generic Services. The PSQL database engine must be manually configured on all nodes as explained in Configure Database Engine Properties with PCC.

Preliminary Requirements

It is essential that Failover Clustering functions correctly before you add the PSQL services. For example, verify that the failover completes and that all resources are available. Complete the action again and failover back to the original node. Refer to the Microsoft documentation for how to set up Failover Clustering, verify it is working correctly, and perform tasks with it.

Just as you would for any service, set up the essential clustering components before you add the PSQL services. As you set up the components, check the following:

- Modify the NTFS user permissions, if required, for the shared disk where you want the PSQL data to reside. The PSQL transactional and relational services typically run under the Local System Account. Ensure that the Local System Account has permissions to read and write to the shared disk.
- Select “SMB” for the share protocol and modify the SMB permissions if required.
- Skip DFS Namespace Publishing. DFS is not required for the PSQL services.

After setting up the components, a good practice is to verify that the machines on which the PSQL Client is installed can communicate with the Client Access Point. Also, verify that the machines can browse to the share name.

**Recommended Installation Process**

The following table explains the recommended process to add PSQL to Cluster Services on Windows Server 2008. The process for Windows Server 2012 is essentially the same with differences noted above.

Table 31  Adding PSQL to Cluster Services on Windows Server (2008 and Later)

<table>
<thead>
<tr>
<th>Action</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install PSQL on the Cluster Nodes</td>
<td>Install PSQL Server on each cluster node and choose identical options for each installations. Do <strong>not</strong> install PSQL on the cluster shared storage, where the PSQL data resides. After installation, the PSQL transactional service and the relational service are both set to start automatically when the operating system starts. Change the startup type to manual. The PSQL transactional and relational services typically run under the Local System Account. Check the domain permissions for your cluster node accounts if you encounter problems installing PSQL.</td>
</tr>
</tbody>
</table>

| Add a Cluster Resource for PSQL and Set the Properties | The transactional service of PSQL is always required as a cluster resource. The relational service is optional and is needed only if your application uses the Relational Engine. Include the following as you specify properties for the cluster resource:  
  - Select the option **Use network name for computer name**.  
  - For the transactional service, use `Software\Pervasive Software` for the Root Registry Key. If you include the relational service, decide how you want to handle data source names (DSNs) for Registry Keys. Perform one of the following:  
    - Specify the key `SOFTWARE\ODBC` if you want to affect all ODBC data sources and ODBC providers installed on the cluster node. Add the following key depending on the operating system architecture: `SOFTWARE\ODBC\ODBC.INI` for 32-bit Windows or `Software\Wow6432Node\ODBC\ODBC.INI` for 64-bit Windows  
    - To specify separate DSNs, do **not** add any key(s) for the registry. Instead, you must first configure the database engines with PCC (see Configure Database Engine Properties with PCC). After that, create the DSNs on an active cluster node, initiate failure, and create the same DSNs on the surviving node. Repeat this until all cluster nodes contain the DSNs. **Note:** Because of the dependencies, bring the PSQL resources online in the following order (and stop them in the reverse order): first the PSQL Transactional Engine then the PSQL Relational Engine. |

*Note:* Because of the dependencies, bring the PSQL resources online in the following order (and stop them in the reverse order): first the PSQL Transactional Engine then the PSQL Relational Engine.
## Ensure the Shared Storage Has the Necessary Files and Directories

The PSQL transactional and relational services typically run under the Local System Account. Ensure that the Local System Account has permissions to read and write to the shared disk.

Copy the DBNAMES.CFG file from the Application Data area on the active node where you installed PSQL to a directory of your choice on the shared storage.

Copy the following directories from the Application Data area on the same node to the same directory on the shared storage. For convenience, you can copy them to the same directory as DBNAMES.CFG, but that is optional.

- defaultdb
- Demodata
- tempdb
- Transaction Logs

If you want to use PSQL System Analyzer to perform transactional or relational tests, also copy the PSQL\Samples directory.

## Configure Database Engine Properties with PCC

You configure the database engine with PSQL Control Center (PCC) to add certain configuration settings to the registry. Configure the engine on the active node in your cluster, then initiate a failure to migrate the settings to the next node(s) in your cluster.

In PCC, set the following engine properties for **Directories**. When PCC prompts you to restart the services, select **No**.

- For **Transaction Log Directory**, specify the location on the shared disk where you copied the Transaction Logs directory.
- For **DBNames Configuration Location**, specify the location on the shared disk where you copied the DBNAMES.CFG file.

In the Failover Cluster Management console, take the PSQL resources offline and back online.

In PCC, set the following engine properties for **Databases**:

- For **DEFAULTDB**, set **Dictionary Location** to the location on the shared disk where you copied the defaultdb directory. For **Data Directory**, add the location on the shared disk where you copied the defaultdb directory and remove the default data directory.
- Set Dictionary Location and Data Directory for databases DEMODATA and TEMPDB to the location on the shared disk where you copied them. For DEMODATA, specify “Demodata” for the Dictionary Location and Data Directory. For TEMPDB, specify “tempdb” for both.

## Migrate the Database Engine Configuration Settings

Migrate the database engine configuration settings to the other node(s) in the cluster. In the Failover Cluster Management console, perform the action **Move the service or application** to move the PSQL service to another node in the cluster.

Verify that the failover completes and that all resources are available. Continue to move the service to each desired node and verify that all resources are available until you failover back to the original node.

PSQL is now configured for a Failover Cluster.

---

### Linux Heartbeat

The Heartbeat program is one of the core components of the Linux-HA (High-Availability Linux) project. Heartbeat runs on all Linux platforms and performs death-of-node detection, communications and cluster management in one process.

This topic discusses adding the PSQL services to Linux Heartbeat and assumes the following:

- You know how to install and configure the Heartbeat program and need only the information required to add PSQL to a Cluster Service group.
You are familiar with using PSQL and its primary utilities such as PSQL Control Center (PCC).

Preliminary Requirements
It is essential that Linux Heartbeat be functioning correctly before you add PSQL to the cluster. Refer to the documentation from the High Availability Linux Project (www.linux-ha.org) for how to install Heartbeat, verify it is working correctly, and perform tasks with it.

Just as you would for any application, set up the essential clustering components before you add PSQL.

Recommended Installation Process
The following table explains the recommended process to add PSQL to Linux Heartbeat.

<table>
<thead>
<tr>
<th>Table 32 Adding PSQL to Linux Heartbeat</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Action</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Install PSQL on the Cluster Nodes</td>
<td>Install PSQL Server on each cluster node and choose identical options for each installations. Do not install PSQL on the cluster shared storage, where the PSQL database(s) resides. After installation, the database engine is set to start automatically when the operating system starts. With clustering, however, Linux Heartbeat controls starting and stopping the database engine. The controlling node in the cluster starts the engine, the other nodes do not. After you install PSQL Server, ensure that the Group IDs for “pvsw” and “pvsw-adm” and the UID for “psql” match on all nodes. If required, change the IDs to ensure they are the same.</td>
</tr>
</tbody>
</table>
| Configure the Shared Storage    | The shared storage is where the PSQL database resides. Shared storage for Heartbeat can be implemented many different ways. The multitude of possible implementations is beyond the scope of this document. This section assumes that an NFS mount is being used. Create (or at least identify) a location on shared storage where you want the database to reside. The location is your choice. Ensure that user psql has read, write, and execute authority for the location. Create two groups and a user on the shared storage to ensure that each cluster node can access the database files.  
  - Groups pvsw and pvsw-adm must match pvsw Group ID and pvsw-adm Group ID, respectively, on the cluster nodes.  
  - User psql must match psql UID on the cluster nodes. |
| Create the Directory for the Shared Storage Mount | On each cluster node, log in as user psql then create a directory that will be mounted to the shared storage. (User psql has no password and can only be accessed through the “root” account with the su command.) The name of the directory is your choice. |
| Configure Heartbeat Server      | Configure the Heartbeat server on each of the nodes that will control the PSQL database engine. Configure the following:  
  - Nodes. Add all nodes that you want in the cluster.  
  - Authentication. Specify the type of authentication to use for the network communication between the nodes.  
  - Media. Specify the method Heartbeat uses for internal communication between nodes.  
  - Start-up. Specify the setting for when the Heartbeat Server starts. Set this to on, which means that the “server starts now and when booting.” |
| Assign Password for Heartbeat User | Linux Heartbeat provides a default user named “hacluster” for logging in to the Heartbeat Management Client. Assign a password to user “hacluster” on each of the nodes from which you want to run Heartbeat Management Client. |
Add a Resource Group for PSQL

Log in as root and start the Heartbeat Management Client on one of the cluster nodes. Log in as user "hacluster" and add a new group. For ID, specify a name for the PSQL group. Set Ordered and Collocated to “true.”

Add the Resources to the Group

Add three resources to the PSQL group:
- IPaddr
- Filesystem
- Psql (OCF resource agent)

IPaddr

In the Heartbeat Management Client, add a new “native” item. For Belong to group, select the group you added for PSQL. For Type, select “IPaddr.”

On the resource you just added, specify the IP address of the cluster for the IP Value. Use the IP address assigned to the cluster (not the node) when Linux Heartbeat was installed and configured.

Filesystem

Add another new “native” item. For Belong to group, select the group you added for PSQL.

For Type, select “Filesystem” and delete the parameter “fstype,” which is not required. Add a new parameter and select “device” for Name. For Value, specify the device name of the shared storage, a colon, and the share mount location.

Add another new parameter and select “directory” for Name. For Value, specify the directory to use with the NFS mount.

Psql (OCF resource agent)

Add another new “native” item. For Belong to group, select the group you added for PSQL. For Type, click “psql” with a Description of “PSQL OCF Resource Agent.” No additional parameters are required for the parameter.

Create the Subdirectories on the Mounted Shared Storage

Now that you have added the Filesystem resource, the mount exists between the cluster server and the shared storage. On one of the cluster nodes, log in as user psql. Under the shared storage mount, create a directory named “log” and another named “etc.”

For example, if the mount directory is “/usr/local/psql/shared,” you would add directories /usr/local/psql/shared/log and /usr/local/psql/shared/etc.

Configure the Cluster Server in PCC

On each of the cluster nodes, you need to configure the cluster server with PSQL Control Center (PCC).

Place all cluster nodes into standby mode except for the one from which you will run PCC. As user psql, start PCC on the one active node or from a client that can access the active node.

In PSQL Explorer, add a new server and specify the name (or IP address) of the cluster.

Access the properties for the server you just added. If prompted to log in, log in as user “admin.” Leave the password blank. Access the Directories Properties. For Transaction Log Directory, specify the directory that you created for the “log” location. For DBNames Configuration Location, specify the directory that you created for the “etc” location. See Create the Subdirectories on the Mounted Shared Storage.

Use PCC to add a new server and set its properties from each of the other cluster nodes. Place all nodes into standby mode except for the one from which you run PCC.
Failover Clustering

Managing PSQL in a Cluster Environment

After you install PSQL in a failover cluster environment, you can manage PSQL as a resource. The following items discuss common management topics.

- PSQL Licensing and Node Maintenance
- PSQL Failure Behavior
- Stopping or Restarting the PSQL Transactional Service
- PSQL Configuration Changes
- Cluster Environment with a Proxy Server
- Software Upgrades

### PSQL Licensing and Node Maintenance

The normal procedure pertaining to PSQL licensing and machine maintenance also applies to the nodes in a failover cluster environment. Deauthorize the PSQL key before you modify the configuration of the physical or virtual machine on which the database engine is installed. Reauthorize the key after the changes are complete.

See [To Deauthorize a Key](#) and [To Authorize a Key](#) in PSQL User's Guide.

### PSQL Failure Behavior

If a cluster node fails, a PSQL client does not automatically reconnect to the PSQL engine on the surviving node. Your application must reconnect the client to the PSQL database or you must restart the application. This applies even if Enable Auto Reconnect is turned on for the database engine.

If transaction durability is turned off and a failure occurs before a transaction completes, the transaction is automatically rolled back to its state before the transaction began. That is, to the last completed check point. The rollback occurs when the active server requests access to the data file.
If transaction durability was turned on, completed changes can be recovered that occurred between the time of the cluster node failure and the last checkpoint. Transaction durability must be configured the same way on all nodes and the transaction log located on the shared storage. Transactions that had not completed at the time of the cluster failure, however, are lost even if transaction durability was in effect.

**Stopping or Restarting the PSQL Transactional Service**

A cluster failover occurs from the active node if you manually stop the PSQL transactional service through the operating system. If you are performing service node maintenance and want to avoid such a failover, stop the PSQL transactional service through the cluster utilities.

**PSQL Configuration Changes**

*Configuration Changes that Require Database Engine Restart*

Some configuration changes require that you restart the database engine. For such changes, perform the changes on an inactive mode. Otherwise, the restart could cause a failover to occur. See *Configuration Reference* chapter.

*Adding a User Count Increase (UCI) Key*

You can add an increase key to PSQL at any time on either the active node or an inactive node. Adding a UCI key does not require an engine restart and takes effect immediately. See also *Increase User Count or Data In Use* in PSQL User's Guide.

**Cluster Environment with a Proxy Server**

If your cluster environment also incorporates a proxy server, see *Authorization Access Through A Proxy Server* in PSQL User's Guide.

**Software Upgrades**

At some point, you may need to upgrade PSQL or the failover cluster software. For PSQL, ensure that you upgrade the database engine on an inactive node. See also the release notes for the upgrade version. For the failover cluster software, refer to the recommended procedure from the software vendor.
Migration

Migration moves a VM running PSQL from one physical host to another. The memory, storage, and network connectivity of the VM are typically migrated to the destination. Depending on the hypervisor, migration is sometimes referred to as “live” migration or “hot” migration.

With a “live” or “hot” migration, client connections to PSQL remain intact. This allows changes to hardware or resource balancing. With a “cold” migration, network connectivity is interrupted because the VM must boot. Client connections to PSQL must be reestablished.

A migration environment has only one instance of PSQL running, which makes the environment somewhat vulnerable if the host machines crashes or must be quickly taken offline. Also, if the shared storage fails, the database engine cannot process reads from or writes to physical storage. Some hypervisors offer a migration solution that does not use shared storage.

As long as host names remain the same after the VM migrates, PSQL continues to operate normally. The product key remains in the active state.

No special steps are required to install or configure PSQL in a migration environment. Refer to the hypervisor documentation.
 Fault Tolerance

A fault tolerant environment is similar to a migration environment but includes additional features to ensure uninterrupted operation even after the failure of a component. A fault tolerant environment ensures network connections, continuous service, and data access through synchronized shared storage. If a component switch occurs, client machines and applications continue to function normally with no database engine interruption.

No special steps are required to install or configure PSQL in a fault tolerant environment. Refer to the hypervisor documentation.
Disaster Recovery

Disaster recovery includes data recovery and site recovery. Data recovery is how you protect and restore your data. Site recovery is how you protect and restore your entire site, including your data.

Data recovery is facilitated with the hypervisor shared storage and PSQL transaction logging and transaction durability. See Transaction Logging and Durability. You can use transaction logging and transaction durability with PSQL Server and Vx Server.

Site recovery can be accomplished with both physical machines and virtual machines. PSQL operates normally provided that host names remain the same in the recovered site. This is typically the case for virtual machines. If you are recovering physical machines and the host name at the recovery site is different, the PSQL product keys will change to the failed validation state when PSQL starts. PSQL will continue to operate normally in the failed validation state for several days, during which you can either repair the key or move back to the original site.

No special steps are required to install or configure PSQL in a disaster recovery environment. Refer to the hypervisor documentation.
PSQL and Hypervisor Products

Tips for Using PSQL with Various Hypervisor Products

This chapter provides topics to help you use PSQL most effectively with a hypervisor product. The chapter contains the following sections:

- Hypervisor Product Installation
- Usage Topics for PSQL
Hypervisor Product Installation

As with any complex software, hypervisor products can be installed various ways, including with nonstandard and atypical configurations. For compatibility with PSQL, install and configure the hypervisor product using the best practices recommendations of the product vendor.
Usage Topics for PSQL

This section discusses the following topics for using PSQL:

- Physical Machine To VM Migration
- Configuration
- VM Resource Pools and Templates
- Failover Cluster Support
- Performance
- Data Backup

Physical Machine To VM Migration

You can install a PSQL engine, PSQL Server, PSQL Vx Server, or PSQL Workgroup, on physical machines initially and transition to VMs as your business needs change. For migrations from physical machines to VMs, you must ensure that the hostname remains the same.

The PSQL engine does not depend on IP address, but the VM itself might. If your VMs depend on raw IP addresses, or on the hosts file, rather than on the Domain Name System (DNS), ensure that you also take appropriate actions concerning IP addresses.

Configuration

No special steps are needed to configure PSQL to use hypervisor product features such as live migration, fault tolerance, high availability, paravirtualization, resource scheduling and disaster recovery. PSQL remains authorized and fully functional provided that the hostname remains consistent.

Certain scenarios, such as for disaster recovery, may require network and hardware changes. You may change the following without adversely affecting PSQL:

- IP or MAC address of the VM
- Hardware in the VM, such as CPU type, CPU speed, amount of memory, and type and size of storage.

Note that PSQL is not aware of certain hardware changes, such as increasing memory or physical storage, if the database engine is running. You must stop and restart the database engine if you want it to be aware of such changes.

VM Resource Pools and Templates

PSQL may be used with VM resource pools and templates. For both uses, each copy of PSQL requires its own product key. See License Enforcement in PSQL User's Guide.

Resource Pools

PSQL must be authorized in each VM within a resource pool that includes the database engine.
Templates
To authorize PSQL in a VM launched from a template, you may use a configuration script. The script can invoke the CLI License Administrator utility to authorize the PSQL key during the customizing of the guest operating system. See License Administrator Command Line Interface in PSQL User's Guide. Remember to customize other properties of the guest operating system, such as hostname, that are independent of running PSQL.

Failover Cluster Support
As a general guideline, if you use affinity rules, ensure that all cores are running on the same socket. This aids performance of PSQL because of its multi-core support. Anti-affinity rules may also be used depending on your configuration.

If you use Raw Device Mapping (RDM) as the data drives for MSCS configurations, be aware of the considerations. Refer to the vendor documentation for RDM.

If you use fault tolerance/high availability with Distributed Resource Scheduler (DRS), be aware that load balancing can be done only after failover. Refer to the vendor documentation for DRS.

Performance
To achieve the best performance for PSQL, ensure the following:

- Adherence to the performance best practices recommendations from the hypervisor vendor.
- The VM hosting PSQL has ample memory (RAM).
- The hypervisor host has enough virtual CPUs to minimize virtual CPU contention among all of the VMs on the same machine. This prevents contention with the VM running PSQL. If you use affinity rules, ensure that all cores are running on the same socket.
- The PSQL data files reside on very fast physical storage and minimize spindle contention and controller contention for the physical storage device.

Data Backup
See Data Backup with Backup Agent and VSS Writer.
This chapter explains how to get the most out of your Workgroup engine.

- Networking
- Technical Differences Between Server and Workgroup
- Troubleshooting Workgroup Issues
- Redirecting Locator Files
Networking

Both the server and workgroup products are shipped with the same networking components. So you can upgrade a workgroup engine to a server engine. The client side requestors can connect to either type of engine.

NetBIOS

The PSQL Network Services Layer searches for an engine on the network using NetBIOS as well as the other protocols previously supported; Named Pipe, DNS, and NDS. The NetBIOS and DNS protocols can be used to contact a workgroup engine. Server engines on Windows advertise themselves with Named Pipes, so NetBIOS is not needed.

Once the client requestor has found an IP, SPX or NetBEUI address, it will try to establish a connection to a MicroKernel engine using that transfer protocol.

MicroKernel Router Decision Algorithm

The client side MicroKernel router discovers Gateway ownership of files by following a well-established algorithm.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Procedure</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Try to connect to a database engine on the same computer as the data files.</td>
</tr>
<tr>
<td>2</td>
<td>Try to open the data files using the local engine (on the client machine).</td>
</tr>
<tr>
<td>3</td>
<td>Find and connect to the Gateway engine that owns the files.</td>
</tr>
</tbody>
</table>

The first thing the client-side router always tries is to connect to an engine on the same computer as the data. Because of this procedure, it is always more efficient to have an engine running where the data is.

Because the PSQL Network Services Layer uses so many different methods to find and connect to a remote database engine, there may be a time delay on the first attempt to open a file on a file server that does not have a database engine running. If Gateway Durability is turned on, that connection will not be attempted thereafter because the router remembers each machine on which it fails to locate an engine.

If the router cannot connect to an engine on the remote file server, the router then allows the local engine to attempt to open the remote files. The local engine first attempts to create a new locator file and take ownership of the remote directory. If the directory is already owned by another MicroKernel, the local engine returns Status Code 116 to the router.

Finally, the router attempts to discover the Gateway computer. It opens the locator file and reads the name of the Gateway engine. Then it sends the request to that engine. Notice that the router never tries to read a locator file unless it has received Status Code 116 from a MicroKernel first. This behavior means that in order to use the Gateway feature, you must have a local workgroup engine installed. If the attempt to open the remote files with the local engine fails because there is no local engine, the router does not try to read the locator file and no Gateway engine is found.
Technical Differences Between Server and Workgroup

The Server Engine and Workgroup Engine have a few significant differences that this section explains.

**Platforms**

The Server Engine has both 32-bit and 64-bit editions available, both on Linux and Windows. The Workgroup Engine has only a 32-bit Windows edition available. Although the Workgroup Engine can be run on a 64-bit Windows operating system, it is restricted to an addressing space of only 4GB and cannot take advantage of the larger amounts of memory typically installed on such systems. When the Workgroup Engine is installed on a 32-bit operating system, it has an addressing space limit of 2 GB.

**User Interface**

The Server Engine for Windows is installed to run as a Windows Service. The Workgroup Engine can be installed to run as an application or as a service. By default, it is installed to run as a service for a fresh install. See Configuring the Workgroup Engine in Getting Started with PSQL. If installed to run as an application, the Workgroup Engine uses a tray icon for an interface.

**Authentication and Btrieve Security Policies**

The Server Engine enforces file permissions set up in the operating system. The Workgroup Engine does not authenticate users on its own. If the Workgroup Engine can access the computer on the network, it can get to the data. This relaxed security is intended for small offices where security is less of an issue and ease of use is more important.

The lack of operating system authentication with the Workgroup Engine means that the Mixed security policy for Btrieve is the same as the Classic security policy. See Security Models and Concepts in Advanced Operations Guide. This difference in security policy is a behavior difference between the Server and Workgroup Engines.

**Gateway Support**

The Workgroup Engine creates locator files everywhere it opens files, both locally and remotely, allowing the Engine to dynamically adjust gateway ownership daily. By default, the Workgroup Engine also runs under a user ID, which can be authenticated on other computers and network devices. This makes the Workgroup Engine ideal for use in a gateway environment. See Setting Up a Gateway Configuration in Getting Started with PSQL.

The Server Engine does not always create or honor gateway locator files. As such, it is not designed or tested for use in a gateway environment. Therefore, replacing a Workgroup Engine with a Server Engine as a gateway in a workgroup environment is not supported.

**Asynchronous I/O**

The Server Engine for Windows makes use of Asynchronous I/O. Furthermore, coalescing of database page writes is done only by the Server Engine. These features can provide a significant performance advantage for the Server Engine over the Workgroup Engine during heavy I/O usage.
Default Configurations

The default values for some database settings (such as cache size and system cache) are different between Server Engine and Workgroup Engine. The default values for Workgroup Engine settings are set to consume less system resources. See Configuration Reference in Advanced Operations Guide.

License Model

PSQL Server and PSQL Workgroup use a user count license model. PSQL Vx Server uses a capacity-based license model. See License Models in PSQL User's Guide.
Troubleshooting Workgroup Issues

This section provides a few tips on troubleshooting problems in a Workgroup environment.

Time delay on first connection

If you are regularly experiencing a delay when the first file-open request is issued, see if these techniques help.

If possible, make sure there is an engine running where the data is

Connecting to an engine on the same machine as the data is the client's first priority when deciding where to send a file-open request. To ensure a Workgroup Engine is running as an application, put an engine icon into the startup folder with the command:

W3dbsmgr.exe

Another option is to install the Workgroup Engine as a service. See Getting Started with PSQL. Also, for default locations of PSQL files, see Where are the PSQL files installed? in Getting Started with PSQL.

If you are running a gateway topology

If you cannot run an engine where the data is, then the time delay during the first connection is a more important issue. Here are a few things you can do.

1 Reduce the supported protocols in the client settings so that protocols that are not used in your network are not attempted.

2 Use Gateway Durability. Gateway Durability is a client configuration setting that allows you to virtually eliminate the delay in making the first connection in a gateway environment. If Gateway Durability is turned on, it forces the client router to write into the registry the names of computers it finds that do not have an engine running. Once a failure to connect happens, instead of remembering this server name only while the router is running in-process, it saves the name in the registry. The next time the application starts up, it does not try to connect to the engine where the data is. It immediately goes to the next step of determining the identity of the current Gateway.

You can turn this setting on in PCC. Within PCC PSQL Explorer, expand Local Client node then right-click MicroKernel Router. Click Properties then click Access. Click the Gateway Durability option to set it to “on” (a check mark indicates that the setting is “on”) then click OK.

Note This feature is OFF by default since it fixes the topology. If you add a server engine or a Workgroup engine where the data is, you must turn this setting back to OFF on each of the clients where you turned it ON. Turning the setting off erases the registry of computers without an engine running, so you can turn it back ON immediately and a new list will be generated based on the new topology.

Status Code 116

Status 116 is a MicroKernel Status Code which means that the file is being used by another MicroKernel engine acting as a Gateway. If your application receives a status code 116, it means that the MicroKernel router can read the locator file but cannot contact the engine running on the gateway computer.
The first thing you need to do is find out who the gateway is. You can perform this task with the Gateway Locator utility.

Next, use PSA network tests to try to connect to that computer. PSA can provide valuable information to isolate the problem.

One situation when this could occur is when the two computers are separated by a router such that they can both see the file server but they cannot see each other.
Redirecting Locator Files

This feature of Gateway engine operation guarantees transaction atomicity for multidirectory databases and also makes it easy to change the name of a Gateway engine across multiple data directories.

Recall that the PSQL client uses the following approach to access remote data files:

1. First, attempt to connect to a database engine on the same computer as the data files.
2. Second, if no database engine is available on the remote machine, attempt to use a local engine to take ownership of the remote directory and create a Locator File. If a Gateway Locator File already exists, the local engine is not used.
3. Third, try to use the specified Gateway engine.

It is important to remember that the Gateway configuration only goes into effect when there is no database engine available on the same computer as the data files.

This feature allows a dynamic (floating) Gateway engine while at the same time preserving transaction durability for multidirectory databases on the same volume. This benefit is provided by a new type of Gateway Locator File that points to another Gateway Locator File. The new type is called a Redirecting Locator File. By having Redirecting Locator Files in directories A, B, and C that point to the Locator File in directory D, you can ensure that the Gateway engine specified by the Locator File in directory D services data files in the other directories as well.

Regardless of whether the Locator file in directory D specifies a permanent Gateway or is dynamically created by the first engine to open those files, this architecture ensures that all the specified directories use the same Gateway engine. Likewise, if you decide to change the permanently assigned Gateway engine for several directories, Redirecting Locator Files allow you to do so by changing only one Locator File, rather than all of them. Thus, it is possible to specify that all data files on a given hard drive must use the same Gateway engine, with or without designating a permanent Gateway.

Redirecting Locator File Requirements

The first line of a Redirecting Locator File must start with “=>” and be followed by a path specifying another Locator File, which must be on the same drive. You can use any combination of forward slash and back slash in the path name. All slashes are converted to the type of separator used by the local operating system.

If your specified path ends with a slash, the database engine assumes the default Locator File name (~PVSW~.LOC) and appends it to the path. If the specified path does not end with a slash, the database engine assumes that the path already contains the file name.

The following table lists the ways a Redirecting Locator File path can be specified:

<table>
<thead>
<tr>
<th>Path</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>=&gt;\path_name</td>
<td>Specifies the path from the root of the drive where the current Locator File is stored.</td>
</tr>
<tr>
<td>.\path_name</td>
<td>Specifies the path relative to the current directory.</td>
</tr>
<tr>
<td>..\path_name</td>
<td>Specifies the path relative to the parent directory of the current directory.</td>
</tr>
</tbody>
</table>
You can assign multiple levels of redirection to these Locator Files. For example, you can have the first Locator File pointing to a second Locator File, the second Locator File pointing to a third Locator File, and so on. Each workgroup engine opens each Locator File sequentially, looking for the actual Gateway name. It stops searching once it has found the locator file that does not start with “=>”. The engine then assumes this Locator File specifies the Gateway engine.

Creating Redirecting Locator Files

As with any Locator File, a Redirecting Locator File is a plain text file. You can create Redirecting Locator Files by hand or programmatically. A Redirecting Locator File must be flagged as read-only, or it will be overwritten by the first engine to attempt to access the data files in that directory.

➢ To Create a Redirecting Locator File

1. Open Notepad or a text editor, and open a new text file.
2. Decide where you are going to save the file when you are finished. You will save the file in the same directory as the data files which you want to redirect to another locator file.
   For example, if you want to ensure that the data files in C:\data are accessed by the same Gateway engine as other data files, then you will want to keep in mind the folder C:\data.
3. Type in => and the path name of the next Locator File. Continuing the example from the previous step, if you want the current data files in C:\data to be owned by the Gateway engine specified in the Locator File located in c:\moredata, then you would type the following:
   =>\moredata\ (recommended) or
   =>\moredata\ (not recommended)
   In the first case, you are specifying a relative path from the current directory. In the second case, you are specifying an absolute path from the root of the current drive. In this particular example, both cases resolve to the same target directory.

➢ Note It is strongly recommended that you use relative path names (starting with ./ or ../) in your Redirecting Locator Files, and that you use the same share names on all workstations to access the same data. Following these two recommendations can prevent errors that may occur with network path name resolution over mapped drives.

4. Save the file as ~PVSW~.LOC in the directory where the data files exist that you want to specify a Gateway engine for.
5. Close Notepad or the text editor.
6. Flag the text file as read-only.

➢ To synchronize many data directories on a permanent Gateway

1. Either by hand or by using the Gateway Locator program, create a read-only (permanent) Locator File that does not redirect. It must specify a Workgroup engine to use as the Gateway.
For example, your locator file may specify the computer named “workgroup1” as the Gateway engine, and the file may be located in \DATA\DB1.

2 For each of the other data directories that you want to use the Gateway engine specified in the previous step, you need to create a Redirecting Locator File in that directory. Each Redirecting Locator File must point to the file you created in the previous step.

Continuing the example, each Redirecting Locator File in \DATA\DB2 and \DATA\DB3 would then contain the following text:

```plaintext
=>..\DB1\n```

This causes any engine reading this file to follow the relative path and search the specified directory \DATA\DB1 for another Locator File. In this case, the specified directory contains a Locator File that names “workgroup1” as the Gateway computer.

➢ To synchronize many data directories on a dynamic Gateway

1 Follow the steps above, only in step #1, ensure that the Locator File is writable, not permanently-assigned.

In this case, remember that if no engines are accessing any data files in the redirecting hierarchy, then there will be no Locator File in the target directory. This is normal. The dynamic Locator File is created each session by the first engine to access the data, and the file is deleted when the last user session ends. It is permissible to have Redirecting Locator Files that point to a data directory that has no Locator File in it. In this case, the first engine to open those data files creates the Locator File.

Example

Using the example Locator Files shown in Figure 4, the Redirecting Locator File on the left forces the database engine to go up one directory, then look in the sub-directory `newdir` for another Locator File with the default name (~PVSW~.LOC). This Locator File, in turn, specifies that the Workgroup engine on the computer named `ntserver1` is the correct Gateway engine. As a result, the database engine on `ntserver1` is used to access the data files in the directory `mydir`.

Figure 4 Redirecting Locator File Example
Monitoring Your PSQL Environment

PSQL is designed for ease of use and minimal administration. At times, however, you may want to monitor various conditions or resources in the PSQL environment. For example, you may need to tune portions of your application and monitor various performance aspects pertaining to the database. Or, you may have adjusted PSQL configuration settings for your business needs and want to monitor the changes from the default settings.

This chapter includes the following sections:

- Monitoring Database State
- Monitoring Data File Fragmentation
- Monitoring Performance Counters
- Monitoring License Usage
- Monitoring Database Access
- Reviewing Message Logs
- Receiving Email Notification of Messages
Monitoring Database State

You can monitor the state of your database with the Monitor utility. It allows you to oversee the following:

- Active Files
- MicroKernel Sessions
- Resource Usage
- MicroKernel Communication Statistics
- SQL Active Sessions

If you are new to Monitor, or want an overview of the utility, see Monitor Overview.

Monitor Overview

Monitor is a utility that allows you to systematically observe certain activities and attributes of the database engine. The utility provides information useful for both database administration and application programming diagnostics. It can monitor aspects of both the MicroKernel Engine and the Relational Engine.

Monitor has two interfaces, both of which provide the same functionality:

- **Graphical Interface Monitor** presents information in a series of tabs.
- **Command Line Interface Monitor** uses an executable program that directs the information to a selected location.

Graphical Interface Monitor

PCC integrates a Monitor utility that presents information organized into a series of tabs. The tabs can be rearranged for your convenience, with columns of data that can rearranged and sorted. It presents a snapshot of a particular moment and can be refreshed either manually or automatically.

Using GUI Monitor

Within PCC, you access Monitor from PSQL Explorer.

➢ To access Monitor for a database engine

You can monitor multiple engines at the same time if you choose.

1. Perform one of the following actions:
   - In PSQL Explorer, locate and open the Engines node. Right-click the database engine that you want to monitor, then click **Monitor** on the context menu.
   - Click **Tools > Monitor**. Select a server to monitor. Click **OK**.

GUI Features

The table below the following image explains the user interface components. Click any area of the image for which you want more information.
You can set preferences for the Monitor from either Monitor itself or from PCC. In either utility, select Window > Preferences > Monitor to open the Monitor tab of the Preferences dialog box.

There are two types of preferences you can set. First, you can set the layout of the Monitor user interface, so that when you reopen Monitor, its tabs are arranged just as you left them. You can set this individually for each server that you access. Second, you can set the features for any particular grid, for example, column width, sort order, and column order. If you open that particular grid on another server, it follows the same settings. That way, you can easily compare the same grid on different servers.
Setting Refresh Options

Information in the Monitor can be refreshed automatically at a configured interval, as desired, or not at all. Be aware that refreshing too many windows at a short interval may slow performance.

| Automatic refresh | 1. Use the Set Automatic Refresh icon to turn on automatic refresh.  
|                   | 2. Either select File > Set Automatic Refresh Rate or click the Set Refresh Rate icon to open the Set Refresh Rate dialog box.  
|                   | 3. In the Set Refresh Rate dialog box, enter the number of seconds between each refresh and click OK. |
| Refresh as desired | Either select File > Automatic Refresh or click the Refresh icon. |
| No refresh | Use the Set Automatic Refresh icon to turn off automatic refresh. |

Tab Functionality

Tabs can be rearranged, separated, and reaggregated for your convenience. To move a tab, put the cursor on the tab label, hold down the left mouse button, and pull the tab label where you want the tab to be.

Because of the different nature of the data on each tab, different operations can be performed on each tab. Those operations are initiated by the icons that appear at the right end of the row of tab labels. The following table describes the icons.

<table>
<thead>
<tr>
<th>Table 35 GUI Icons</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Icon</strong></td>
</tr>
<tr>
<td>Set Automatic Refresh</td>
</tr>
<tr>
<td>Set Refresh Rate</td>
</tr>
<tr>
<td>Refresh</td>
</tr>
<tr>
<td>Select Columns to Display</td>
</tr>
<tr>
<td>Hide/Show Handles</td>
</tr>
</tbody>
</table>
| Delete Selected Session | Removes a highlighted session from the tab. This icon becomes activated only after a session is selected.  
| Caution: This procedure actually terminates a session, so you are interrupting someone's work in progress. Consequently, a message asks you to confirm that you actually want to do this. |
Monitoring Database State

Monitoring Active Files

The Active Files tab provides information about MicroKernel files that are currently open. To select columns to monitor, see Select Columns to Display.

Table 36  Selectable Columns for Monitoring MicroKernel Active Files

<table>
<thead>
<tr>
<th>Path</th>
<th>Provides the directory and all subdirectories to the location of the file.</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Indicates the name, including suffix, of the file.</td>
</tr>
<tr>
<td>Page Size</td>
<td>Indicates the size in bytes of each page in the file.</td>
</tr>
<tr>
<td>Read-Only</td>
<td>Indicates whether the file is flagged as read-only by the operating system.</td>
</tr>
<tr>
<td>Record Locks</td>
<td>Indicates whether any of the active handles for the selected file have record locks. Any application can read a locked record, but only the application that placed the lock can modify or delete the record. A record lock exists only as long as the application that opened the file is updating a record. “Yes” indicates that one or more record locks are applied to the file. “No” indicates that no records are locked.</td>
</tr>
<tr>
<td>Transaction Lock</td>
<td>Indicates whether any of the active handles for the selected file have a transaction lock. A transactional file lock exists only as long as the application that opened the file is processing a transaction.</td>
</tr>
<tr>
<td>Physical File Size</td>
<td>Indicates the size of the file in kilobytes (KB). This information is particularly useful for the capacity-based license model if you want to review data in use on a file-by-file basis. See also Capacity-based License Model in PSQL User’s Guide. Monitor uses kilobytes (KB) for the size of an individual file and megabytes (MB) as the units for resource usage (Monitoring Resource Usage). License Administrator uses gigabytes (GB) as the units because that is how data in use is associated with a key. The different contexts require units appropriate for each context. If a file is immediately closed after you insert a large number of records, Monitor does not immediately reflect the changes in file size. For example, the statistics for “Physical File Size KB” are not refreshed for that file until the next time the file is opened for reading or writing.</td>
</tr>
</tbody>
</table>

You can view the handle information for any active file. See Hide/Show Handles. Active file handles include the following information.

Table 37  Selectable Columns for Active File Handles

<table>
<thead>
<tr>
<th>Client</th>
<th>Indicates the name (typically the login ID of the user) or an index to the Client list of the database server.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Number</td>
<td>Displays the network connection number of the client. If the client does not have a network connection, this field displays “NA” for “not applicable”</td>
</tr>
</tbody>
</table>
Table 37  Selectable Columns for Monitoring MicroKernel Sessions

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Displays the process-supplied task number for processes originating at the server or a Windows Client.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site</td>
<td>Specifies the location of the user process (local or remote).</td>
</tr>
<tr>
<td>Network Address</td>
<td>Identifies the location of the calling process on the network. If the calling process is SPX, the network node/network address is preceded by S: such as S: 65667678 00000000001. If the calling process is TCP/IP, the address is preceded by T4 for IPv4 addresses, T6 for IPv6 addresses, and T for the fully qualified domain name of a client machine. Examples: T4: 180.150.1.24 T6: 1234:5678:0000:0000:0000:0000:9abc:def0 T: &lt;mymachine.mydomain.mycompany&gt;.com</td>
</tr>
<tr>
<td>Open Mode</td>
<td>Indicates the method the application uses to open the specified handle of the file. Valid open modes are the following: • Normal – The application that opened the file has normal shared, read/write access to it. • Accelerated – The application that opened the file has shared read/write access. • Read-only – The application that opened the file has read-only access; it cannot modify the file. • Exclusive – The application that opened the file has exclusive access. Other applications cannot open the file until the calling application closes it. Monitor also specifies all open modes as non-transactional or shared locking when applicable.</td>
</tr>
<tr>
<td>Record Lock Type</td>
<td>Displays the type of record lock(s) currently held by the handle. The possible record lock types are Single, Multiple, and None. Single-record locks enable a user to lock only one record at a time. Multiple-record locks enable a user to lock more than one record at a time.</td>
</tr>
<tr>
<td>Wait State</td>
<td>Indicates whether the user is waiting because of some type of lock on this handle: Waits for Record Lock, Waits for File Lock, or None.</td>
</tr>
<tr>
<td>Transaction State</td>
<td>Displays the state of the transaction lock currently held by the handle. The possible transaction types are Exclusive, Concurrent, or None.</td>
</tr>
</tbody>
</table>

Monitoring MicroKernel Sessions

The MicroKernel Sessions tab provides information about current connections to the MicroKernel Engine. To select columns to monitor, see Select Columns to Display.

Table 38  Selectable Columns for Monitoring MicroKernel Sessions

<table>
<thead>
<tr>
<th>Session</th>
<th>Provides a unique identifier for the connection. A “session” is defined as a client ID used by the MicroKernel Engine or a connection to the Relational Engine. “Client ID” is defined as a 16-byte structure that combines elements provided by the application, by the client platform, and by the database engine to uniquely identify a database transaction context. Session information reflects the sessions established through the MicroKernel Engine and through the Relational Engine. If you want to view sessions established only through the Relational Engine, see Monitoring SQL Active Sessions. This tab allows you to delete a session or all sessions. See Delete Selected Session and Delete All Sessions.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connection Number</td>
<td>Displays the network connection number of the session. If the session does not have a network connection, this field displays “NA” for “not applicable”.</td>
</tr>
</tbody>
</table>
You can view the handle information for any MicroKernel session. See Hide/Show Handles. MicroKernel session handles include the following information.

Table 38 Selectable Columns for Monitoring MicroKernel Sessions

<table>
<thead>
<tr>
<th>Task Number</th>
<th>Displays the process-supplied task number for processes originating at the server, or from a Windows Client.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Site</td>
<td>Specifies the location of the session process (local or remote).</td>
</tr>
<tr>
<td>Network Address</td>
<td>Identifies the location of the calling process on the network. If the calling process is SPX, the network node/network address is preceded by S: such as S: 65667678 0000000001.</td>
</tr>
<tr>
<td></td>
<td>If the calling process is TCP/IP, the address is preceded by T4 for IPv4 addresses, T6 for IPv6 addresses, and T for the fully qualified domain name of a client machine.</td>
</tr>
<tr>
<td></td>
<td>If multiple clients from a single machine connect by different TCP/IP addresses, each address is valid for that client. However, internally to the database engine, an address associated with a client may not be the actual address used by that client. This is because of the way the database engine identifies and manages multiple clients from the same machine. Consequently, since Monitor is reporting engine information, the utility may display an associated address instead of the actual address.</td>
</tr>
<tr>
<td>Locks Used</td>
<td>Indicates the number of locks the session is currently using.</td>
</tr>
<tr>
<td>Transaction State</td>
<td>Displays the type of transaction lock the session currently holds. The possible transaction types are Exclusive, Concurrent, or None.</td>
</tr>
<tr>
<td>Read Records</td>
<td>Displays the number of records read since the session first opened a file.</td>
</tr>
<tr>
<td>Inserted Records</td>
<td>Displays the number of records the session has inserted.</td>
</tr>
<tr>
<td>Deleted Records</td>
<td>Displays the number of records the session has deleted.</td>
</tr>
<tr>
<td>Updated Records</td>
<td>Displays the number of records the session has updated.</td>
</tr>
<tr>
<td>Disk Accesses</td>
<td>Indicates the number of times the session required a disk access. You will not see any information for disk accesses for files that have just been opened.</td>
</tr>
<tr>
<td>Cache Accesses</td>
<td>Indicates the number of times this client finds data in L1 or L2 cache in order to fulfill the request.</td>
</tr>
</tbody>
</table>

You can view the handle information for any MicroKernel session. See Hide/Show Handles. MicroKernel session handles include the following information.

Table 39 Selectable Columns for MicroKernel Session Handle Information

| Path | Provides the directory and all subdirectories to the location of the file. |
| File | Indicates the name, including suffix, of the file. |
Monitoring Resource Usage

The Resource Usage tab displays the resources in use by the MicroKernel since the engine was last started. To select columns to monitor, see Select Columns to Display.

The database engine dynamically controls the maximum values for some of these resources. The maximum value for User Count, Session Count, and Data In Use depends on the product license. See License Models in PSQL User's Guide.

If a resource does not apply to the type of PSQL product being monitored, “n/a” (“not applicable”) appears for each statistic. For example, “User Count” does not apply to PSQL Vx Server. Therefore, “n/a” appears as the Current, Peak, and Maximum value for “User Count” if PSQL Vx Server is being monitored. Similarly, “n/a” appears as the Maximum value for “Session Count” and “Data in Use MB” if PSQL Server is being monitored.

If you are considering using PSQL Vx Server, you need the ability to estimate Current and Peak values for “Session Count” and “Data in Use MB”; consequently, those statistics are displayed for PSQL Server without being enforced. No notifications are sent about them regardless of their values.

<table>
<thead>
<tr>
<th>Engine Uptime</th>
<th>Lists the amount of time in weeks, days, hours, and minutes that the MicroKernel engine has been running. Engine Uptime is not a selectable column for Resource Usage. It is part of the grid title.</th>
</tr>
</thead>
</table>

| Table 39 Selectable Columns for MicroKernel Session Handle Information |
|-------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| **Open Mode**                                   | Indicates the method the application uses to open the specified handle of the file. Valid open modes are:                                                                                       |
| Normal – The application that opened the file has normal shared, read/write access. |                                                                                                                                                    |
| Accelerated – The application that opened the file has shared read/write access. |                                                                                                                                                    |
| Read-only – The application that opened the file has read-only access; it cannot modify the file. |                                                                                                                                                    |
| Exclusive – The application that opened the file has exclusive access. Other applications cannot open the file until the calling application closes it. |                                                                                                                                                    |
| Monitor also specifies all open modes as non-transactional or shared locking when applicable. |
| **Record Lock Type**                            | Displays the type of record lock(s) currently held by the handle. The possible record lock types are Single, Multiple, and None.                                                      |
| Single-record locks enable a user to lock only one record at a time. Multiple-record locks enable a user to lock more than one record at a time. |
| **Wait State**                                  | Indicates whether the user is waiting because of some type of lock on this handle: Waits for Record Lock, Waits for File Lock, or None.                                           |
| **Transaction State**                           | Displays the state of the transaction lock currently held by the handle. The possible transaction types are Exclusive, Concurrent, or None.                                                   |

| Table 40 Selectable Columns for Monitoring MicroKernel Resource Usage |
|---------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| **Resource**                                                       | Indicates the type of resource being monitored. See Resource Types for Resource Usage Monitoring.                                                                                                  |
| **Current**                                                        | Indicates the current usage by a resource.                                                                                                                                                    |
The following table lists the types of resources for usage monitoring.

### Table 41 Resource Types for Resource Usage Monitoring

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Files</td>
<td>Indicates the number of files currently open by the MicroKernel. The maximum for this resource is unlimited.</td>
</tr>
<tr>
<td>Handles</td>
<td>Indicates the number of active handles. The MicroKernel creates a handle each time a user opens a file. A single session can have several handles for the same file. The maximum for this resource is unlimited.</td>
</tr>
<tr>
<td>Clients</td>
<td>Indicates the number of clients accessing the MicroKernel. A machine can have multiple clients accessing the database engine simultaneously. The engine dynamically manages the client list. The maximum for this resource is unlimited (the number of clients is limited only by the memory in the computer). “Client” indicates a session established by a client ID (transactional engine interface) or a connection to the relational engine interface. The database engine uses various client sessions for its own internal processes, such as for accessing PSQL system files, metadata files, dbnames.cfg, and default system databases. The number of clients indicates both internal client sessions and non-internal client sessions (see Monitoring MicroKernel Sessions).</td>
</tr>
<tr>
<td>Worker Threads</td>
<td>Indicates the number of concurrent MicroKernel processes.</td>
</tr>
<tr>
<td>User Count</td>
<td>Indicates the number of concurrently connected users. The maximum value shows the maximum permitted users as granted by a license agreement.</td>
</tr>
<tr>
<td>Session Count</td>
<td>Indicates the number of sessions in use by the database engine. For brevity, “number of sessions in use” is also referred to “session count.” The maximum value (also called the “session count limit”) shows the maximum permitted sessions as granted by a license agreement. Session count reflects all sessions whether established through the MicroKernel Engine or through the Relational Engine. Messages pertaining to session count are logged to the various PSQL logging repositories. See Reviewing Message Logs. The database engine uses various sessions for its own internal processes, such as for accessing PSQL system files, metadata files, dbnames.cfg, and default system databases. These internal sessions do not consume any session counts.</td>
</tr>
</tbody>
</table>
Monitoring MicroKernel Communication Statistics

The MicroKernel Communication Statistics tab displays information about communication with the MicroKernel Engine. It includes separate sections for Communications Statistics and Resource Usage Information. Communication statistics are calculated in terms of total number of occurrences processed since the database engine was started.

To select columns to monitor, see Select Columns to Display.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Indicates the type of resource being monitored. See Resource Types for MicroKernel Communications Statistics.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total</td>
<td>Indicates the total number of occurrences processed since the database engine was started.</td>
</tr>
<tr>
<td>Delta</td>
<td>Indicates the number of occurrences processed since you last accessed the MicroKernel Communications Statistics tab. To restart the count of the delta number, see Reset the Deltas.</td>
</tr>
</tbody>
</table>
Table 43 Resource Types for MicroKernel Communications Statistics

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Total Requests Processed</td>
<td>Indicates the number of requests the database engine has handled from workstations or remote, server-based applications.</td>
</tr>
<tr>
<td>SPX Requests Processed</td>
<td>Indicates the number of SPX requests the database engine has handled from clients or remote, server-based applications.</td>
</tr>
<tr>
<td>TCP/IP Requests Processed</td>
<td>Indicates the number of TCP/IP requests the database engine has handled from clients or remote, server-based applications.</td>
</tr>
<tr>
<td>NetBIOS Requests Processed</td>
<td>Indicates the number of NetBIOS requests the database engine has handled from clients or remote, server-based applications.</td>
</tr>
<tr>
<td>Connection Timeouts</td>
<td>Indicates the number of times Auto Reconnect has timed out when attempting to reconnect to Clients. See also Auto Reconnect Timeout.</td>
</tr>
<tr>
<td>Connection Recoveries</td>
<td>Indicates the number of times the AutoReconnect feature has successfully recovered from a connection timeout.</td>
</tr>
</tbody>
</table>

Resource Usage Information

Resource usage information provides current, peak, and maximum values for resource occurrences.

Table 44 Selectable Columns for Monitoring Resource Usage Pertaining to MicroKernel Communications Statistics

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Resource</td>
<td>Indicates the type of resource being monitored. See Resource Types for Resource Usage Pertaining to MicroKernel Communications Statistics.</td>
</tr>
<tr>
<td>Current</td>
<td>Indicates the current usage by a resource.</td>
</tr>
<tr>
<td>Peak</td>
<td>Indicates the highest value for a resource since the MicroKernel was started.</td>
</tr>
<tr>
<td>Maximum</td>
<td>Indicates the highest value allowed for a resource.</td>
</tr>
</tbody>
</table>

Table 45 Resource Types for Resource Usage Pertaining to MicroKernel Communications Statistics

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Communications Threads</td>
<td>Indicates the number of remote requests that the MicroKernel is currently processing. Local requests are not included in this statistic. For the total number of remote and local threads being processed, see Monitoring Resource Usage. The database engine dynamically increases the number of communications threads as needed up to the maximum allowed. For Windows and Linux platforms, the maximum is 1,024. Worker threads are also used to process Monitor requests, so you might not see the number of current worker threads drop below one. This is normal.</td>
</tr>
<tr>
<td>Total Remote Sessions</td>
<td>Indicates the number of remote clients connected to the database engine. The maximum number is dynamic and displays as zero.</td>
</tr>
<tr>
<td>SPX Remote Sessions</td>
<td>Indicates the number of remote clients connected through the SPX protocol to the database engine.</td>
</tr>
</tbody>
</table>
Monitoring SQL Active Sessions

The **SQL Active Sessions** tab provides information about current connections to the Relational Engine. This tab also allows you to delete a SQL session. See **Delete Selected Session**. To select columns to monitor, see **Select Columns to Display**.

### Table 46  Selectable Columns for Monitoring SQL Active Sessions

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name</td>
<td>Provides the login name of the user.</td>
</tr>
<tr>
<td>Client Host Name</td>
<td>Identifies the name of the Client machine for the selected User Name. If unavailable, this is set to “Unknown.”</td>
</tr>
<tr>
<td>Network Address</td>
<td>Identifies the Client machine’s IP or SPX address for the selected User Name. If unavailable, this is set to “Unknown.” Values displayed include IP, SPX, Shared Memory and Unknown.</td>
</tr>
<tr>
<td>Client Application</td>
<td>Identifies the connected application or module. If unavailable, this is set to “Unknown.”</td>
</tr>
<tr>
<td>Data Source Name</td>
<td>Identifies the name of the DSN referenced by the Client application.</td>
</tr>
<tr>
<td>Connection Status</td>
<td>Specifies the connection status for the selected User Name. A status can be any of the following:</td>
</tr>
<tr>
<td></td>
<td>• Active – The session has files open, and that Idle means that the session has no files open.</td>
</tr>
<tr>
<td></td>
<td>• Idle – The session has no files open.</td>
</tr>
<tr>
<td></td>
<td>• Dying – A temporary status that indicates an active session has been deleted but has not finished processing the SQL code. At a suitable termination point, the session is no longer listed on the SQL Active Session dialog.</td>
</tr>
<tr>
<td></td>
<td>• Unknown – Status is unavailable.</td>
</tr>
<tr>
<td>Active/Idle Period</td>
<td>Displays the duration of time, in milliseconds, since the connection has been active or idle.</td>
</tr>
<tr>
<td>Total ConnectionTime</td>
<td>Displays the duration of time, in seconds, since the connection has been established</td>
</tr>
</tbody>
</table>

**Command Line Interface Monitor**

The command line interface (CLI) version of Monitor provides the same monitoring functionality as the GUI version.

This subsection contains the following topics:

- Accessing CLI Monitor
- Configuration File
- Monitoring Output

**Accessing CLI Monitor**

CLI Monitor runs on the Windows and Linux platforms supported by PSQL:
On Windows, the executable program is `bmon.bat` and is installed, by default, in the `\bin` directory of the PSQL installation directory. See Where are the PSQL files installed? in Getting Started With PSQL.

On Linux, the executable program name is `bmon` and is located, by default, in the `/usr/local/psql/bin` directory. Certain requirements must be met before you can run bmon on Linux. These requirements are the same as for another Java utility, bcfg. See Requirements for Running bcfg on Linux, and Troubleshooting Guide for Running bcfg on Linux.

**Configuration File**

Bmon requires a configuration file to provide its settings. PSQL provides a sample configuration file named `monconfig.txt`. It is located, by default, in the `\bin` directory of the PSQL installation directory. See Where are the PSQL files installed? in Getting Started With PSQL.

Refer to the comments in the sample configuration file for the settings that you can configure.

**Monitoring Output**

Output from Bmon can be directed to the console, a log file, or both. An application could, for example, check for a particular condition from the console or in a log file, then take appropriate action.

The configuration file specifies where to direct the output.

**Command Syntax**

```
bmon -f [filepath]config_file [-runonce]
```

**Options**

- `-f` This is a required parameter to specify that a configuration file is providing input to the utility.
- `filepath` The path to the configuration file. If omitted, Bmon checks the local directory for the configuration file.
- `config_file` The name of the configuration file. The file name is of your choosing.
- `-runonce` This is an optional parameter that instructs the utility to run once, then exit. The runonce parameter is particularly useful when Bmon is used in a batch file.

See also Keyboard Key Response If Runonce Parameter Omitted.

**Keyboard Key Response If Runonce Parameter Omitted**

The runonce parameter is optional. If omitted, the utility executes the settings in the configuration file, then pauses for the duration of the refresh rate. During the pause, you can send the utility a valid keyboard key response as shown in Bmon Refresh Rates and Keyboard Key Responses.
If the refresh rate is set to zero, the utility pauses indefinitely until it receives a valid keyboard key response. The `refreshrate` setting in the configuration file specifies how many seconds to pause. By default, `refreshrate` is set to the minimal allowed value of 5 seconds.

Table 47: Bmon Refresh Rates and Keyboard Key Responses

<table>
<thead>
<tr>
<th>Refresh Rate</th>
<th>Keyboard Key Response</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>refreshrate=0</code> (pause until valid keyboard key response received)</td>
<td>Q (or q) + Enter stops execution of bmon</td>
</tr>
<tr>
<td></td>
<td>R (or r) + Enter refreshes the monitoring (runs bmon again)</td>
</tr>
<tr>
<td><code>refreshrate=seconds_to_pause</code></td>
<td>Q (or q) + Enter stops execution of bmon</td>
</tr>
<tr>
<td>where <code>seconds_to_pause</code> is a whole number 5 or greater (pause for <code>seconds_to_pause</code> seconds)</td>
<td></td>
</tr>
</tbody>
</table>
Monitoring Data File Fragmentation

Over time in a busy database as records are created, updated, or deleted, data can become fragmented, lengthening times for file access and transaction response. This fragmentation differs from file system fragmentation on a hard disk because it occurs within the data file itself. As a developer or DBA, you may know when a file is likely to fragment from heavy use, but in some systems, you may be guessing.

Defragmenter is a utility that helps you solve this problem by finding data fragmentation and enabling you to correct it. Defragmentation rearranges records and indexes in data files and removes unused space so that data can be efficiently accessed again. Defragmenting a file does not alter its data in any way, and records can be created, read, updated, or deleted while their files are being defragmented. You can use Defragmenter features during database engine execution with no need for down time or disruption of business operations in most cases.

Opened as a graphical tool in PSQL Control Center, Defragmenter shows data files in use, including their number of reads and writes so that you can quickly find ones under heavy use. You can use drag-and-drop or a button to add files or tables to the Watch List tab. If you know of other items of concern, you can browse to their location and add them to be watched as well.

Defragmenter also runs as the `dbdefrag` utility.

The following topics cover the use of Defragmenter:

- **Deciding When to Defragment**
- **When Defragmenter Cannot Be Used**
- **Accessing Defragmenter**
- **Defragmenter GUI**
  - Setting Defragmenter Preferences
  - Setting Automatic Refresh Interval
  - Arranging Tabs
- **Defragmenter Tasks**
- **Command Line Interface Defragmenter**

**Deciding When to Defragment**

Defragmenter helps you analyze your data files for statistics that may explain loss of performance. High statistics for file size and percentage fragmented, unused, and not in order can explain loss of database performance. By defragmenting a file or table, you can reduce all four of these numbers, which are explained in more detail under the Watch List topic. Transactions generally run more quickly against a newly compacted, reordered file, restoring efficiency, capacity, and performance.

Every database is different, so it isn't possible to make recommendations that apply to everyone. The decision to defragment depends on your knowledge and experience with your own database and its applications. However, some general considerations are worth making:

- Performance remains constant in read-only databases, but as reads and writes occur over time, analysis of watched files will show rising values for statistics. Changes in database behavior may also become noticeable, such as queries and reports running more slowly.
- Bulk delete actions often greatly increase unused space in files, which can be corrected by defragmenting them.
Monitoring

- Read and write operations can continue during defragmentation, but you may see some effect on engine performance. The following considerations may make low traffic times preferable in your defragmentation plan:
  - Defragmentation uses resources that normally are fully devoted to operations on data files, so the extra demands on the engine may be noticeable.
  - Defragmenting must very briefly lock each data file it runs against. In low traffic periods, this lock may not be noticeable, but in high traffic times, client applications may wait longer for a response from the database.

If you do not see performance improvements after defragmenting, then the problem likely lies elsewhere and requires a different diagnosis and solution.

**When Defragmenter Cannot Be Used**

In some cases, Defragmenter cannot be used, or the Rebuild tool should be used instead.

- Defragmentation of files on clients is not currently supported. Files can be defragmented only on the local server where the tool runs.
- Changes to metadata and indexes are not supported during defragmentation. These changes include the following:
  - Creating, modifying, and deleting bound databases
  - Modifying a schema, such as creating, modifying, or deleting a column
  - Creating, modifying, and deleting indexes
  - Modifying or deleting the database, a data file, or a table
  - Modifying or deleting owner names
- Backup Agent operations and continuous operations for data backup cannot be performed on a file undergoing defragmentation.
- If you need to alter data file properties, such as page size, compression, or file version, use Tools > Rebuild instead of Defragmenter.
- Defragmenter stops and exits automatically if it finds corrupted or erroneous records. In these cases, use Rebuild to recover the data. Rebuilt files normally do not need to be defragmented.
- If you are using a client cache engine, and it has already opened a file on the server, then server defragmentation is no longer allowed and requests to defragment return an error. To allow defragmenting, you must restart the server to clear the client connection.
- For DataExchange users: Defragmentation does not change the system data and key used by DataExchange. After defragmenting, you do not need to run the table synchronization and check utility (dxsynctables).
- Defragmentation is not supported for server engines in environments that use Microsoft Volume Shadow Copy Service (VSS) for backup operations.

**Accessing Defragmenter**

Defragmenter runs in two ways:

- In PSQL Control Center, select **Tools > Defragmenter**. For details, see **Defragmenter GUI**.
- At a command prompt, run `dbdefrag`. Note that this command differs from `defrag`, the Microsoft utility for defragmenting the Windows file system. For details, see **Command Line Interface Defragmenter**.
**Defragmenter GUI**

The following screen shot shows the Defragmenter GUI. The table after the image describes the GUI objects. Click an area of the image for which you want more information.

**Figure 6** Defragmenter Graphical User Interface

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File menu</td>
<td>Allows you to set the refresh rate for the Watch List tab, exit Defragmenter, or exit PCC.</td>
</tr>
<tr>
<td>Window menu</td>
<td>Allows you to return to the PCC window or to set preferences.</td>
</tr>
<tr>
<td>Help menu</td>
<td>Allows you to access documentation and logs and to check the PSQL version.</td>
</tr>
</tbody>
</table>

**Table 48** Defragmenter Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Refresh</td>
<td>Allows you to manually refresh the list of items in the Files In Use, Tables, or Btrieve Files tabs. Note that setting the automatic refresh rate affects only the Watch List tab.</td>
</tr>
<tr>
<td>Add to Watch List</td>
<td>Enters a selected item in the Watch List tab.</td>
</tr>
<tr>
<td>Analyze file</td>
<td>Gathers and displays fragmentation statistics for items selected in the Watch List tab.</td>
</tr>
</tbody>
</table>
Monitoring

Table 48 Defragmenter Icons

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cancel defragmentation</td>
<td>Allows you to manually stop a defragmentation action.</td>
</tr>
<tr>
<td>Defragment file</td>
<td>Starts defragmentation for items selected in the Watch List tab.</td>
</tr>
<tr>
<td>Select columns to display</td>
<td>Opens a dialog of check boxes to customize the columns in the Watch List tab. The default is for all columns to be displayed.</td>
</tr>
<tr>
<td>Remove file</td>
<td>Removes an item from the Watch List tab.</td>
</tr>
<tr>
<td>Set the automatic refresh rate</td>
<td>Opens a dialog to set how often the Watch List tab updates progress. The value is in seconds. The default is 5.</td>
</tr>
<tr>
<td>Show or Hide details</td>
<td>Toggles the display of the Details pane.</td>
</tr>
</tbody>
</table>

Defragmenter Tab Views

Defragmenter presents database tables, files, and other objects in tabs in the Defragmenter window.

- **Files In Use**
- **Tables**
- **Btrieve Files**
- **Watch List**

The tabs in the window can be rearranged. See *Arranging Tabs* below.

**Files In Use**

The Files In Use tab lists items currently or recently opened by the PSQL engine. You can add any file shown in this tab to the Watch List tab to monitor, analyze, and defragment. It is best practice to select only files involved in routine database execution, since they are most likely to contribute to performance issues.

The Files In Use tab provides a sorting feature activated by clicking the Reads/Writes column heading. Sorting a long list of files in use helps to identify which files are in heaviest use and possibly more fragmented.

**Tables**

The Tables tab resembles PSQL Explorer, showing nodes for the server where Defragmenter is running. You can add any file shown in this tab to the Watch List tab.

**Btrieve Files**

The Btrieve Files tab is a file explorer for drives and directories on the file system. In addition to data files, this tab also lists items not involved in routine operations, such as dbnames.cfg and .ddf files. It is
possible to add these files to the Watch List tab to analyze and defragment, but unlike data files, attempting to change them while they are in active use returns errors. These metadata objects are at low risk of fragmentation because they generally do not change in a production environment. In the rare case of their needing defragmenting, it can be done during down-time maintenance.

Watch List

Items in the Watch List tab display several types of statistics after an Analyze or Defragment action has been applied.

<table>
<thead>
<tr>
<th>Statistic</th>
<th>Measures</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>File size (MB)</td>
<td>File size in megabytes</td>
<td>The longer the file has been in use and the larger the file, the greater its fragmentation could be. Smaller file sizes typically take less time to defragment unless they have a large number of indexes. For Vx users, the larger size of fragmented files may be a concern, in which case defragmentation may help to stay within your license limit.</td>
</tr>
<tr>
<td>% Fragmented</td>
<td>Percentage of data divided into small blocks that are not contiguous</td>
<td>A lower percentage indicates fewer, larger data blocks stored closer together, allowing more rapid reads and writes. Note: This statistic is not supported for Btrieve 6.x and 7.x file formats.</td>
</tr>
<tr>
<td>% Unused</td>
<td>Percentage of unused space</td>
<td>The lower the percentage, the more compact the file, allowing more rapid reads and writes. Unused space is often created by update and delete operations. Note: This statistic is not supported for Btrieve 6.x and 7.x file formats.</td>
</tr>
<tr>
<td>% Not in Order</td>
<td>Percentage of records not stored sequentially</td>
<td>A lower percentage generally gives higher performance for actions such as table scans on large files. Inserts over time result in records stored in nonsequential order. Note: In some files, the lowest possible percentage not in order may be higher than zero. Further defragmentation does not decrease this statistic because records in the file are already stored as efficiently as possible.</td>
</tr>
<tr>
<td>Last Run</td>
<td>Date and time of last action</td>
<td>Action can be Analyze or Defragment.</td>
</tr>
<tr>
<td>Status</td>
<td>Report from last action</td>
<td>Blank for newly added item. Typically analysis or defragmentation completed.</td>
</tr>
<tr>
<td>Table</td>
<td>Logical location</td>
<td>Database and table or file name</td>
</tr>
<tr>
<td>Path</td>
<td>Physical location</td>
<td>File system path</td>
</tr>
</tbody>
</table>

Details

Under the Watch List tab, a Details pane summarizes the statistics in the Watch List columns, but with some additional information:

- Time taken to defragment the item
- Before and after comparisons of file size and percentage fragmented, unused, and not in order
Monitoring

Selecting an item in the Watch List tab displays its details. If you select more than one item, details are shown for the first one highlighted.

**Setting Defragmenter Preferences**

You can set preferences for Defragment from either the utility itself or from PSQL Control Center. In either GUI, select Window > Preferences > PSQL > Defragmenter to open the Defragmenter tab of the Preferences dialog box.

The following table shows the types of preferences you can set and what they do.

<table>
<thead>
<tr>
<th>Preference</th>
<th>When selected...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remember window layout.</td>
<td>Saves the current arrangement of Defragmenter tabs when you exit the tool, as well as the overall window size, height, and position.</td>
</tr>
<tr>
<td>Don’t warn about incompatible actions during multi-select operations.</td>
<td>Stops the dialog box from showing when actions will be unevenly applied to items of different types.</td>
</tr>
<tr>
<td>Don’t warn about canceling defragmentations again.</td>
<td>Stops the dialog box from showing a warning when you cancel defragmentation.</td>
</tr>
</tbody>
</table>

**Setting Automatic Refresh Interval**

Information in Defragmenter can be refreshed automatically at a configured interval, as desired. The default is to refresh every 5 seconds. Be aware that refreshing too many files too often may slow performance.

**Arranging Tabs**

The Defragmenter tabs can be rearranged and separated for your convenience. To move a tab, drag and drop the tab label where you want the tab to be. If the setting to remember window layout is selected in the Defragmenter preferences, your tab arrangement will be kept.

**Defragmenter Tasks**

This topic provides steps for using features for various tasks in the Defragmenter GUI.

> **To set automatic refresh interval**

The automatic refresh rate applies to the statistics in the Watch List tab. Refreshing of lists in the other tabs is done manually by clicking the Refresh icon or right-clicking the selecting Refresh.

1. Click File > Set Automatic Refresh Rate or click the Set the Automatic Refresh Rate icon to open the refresh rate dialog box.
2. In the dialog box, enter the number of seconds between each refresh. The value must be an integer greater than 1.
3. Click OK.

> **To add a file or table to the Watch List tab**

Adding a file or table to the Watch List tab can be done in a number of ways:
In Defragmenter, select the item in the Files In Use, Tables, or Btrieve Files tabs and then click the Add to Watch List icon at upper right in the tab.

In Defragmenter, right-click an item in the Files In Use, Tables, or Btrieve Files tabs and select Add to Watch List.

In Defragmenter, drag and drop an item from any tab to the Watch List tab.

Items added to the Watch List tab appear at the end, after the items already there.

> To analyze a file or table

Analysis reports statistics that determine which items may need defragmenting.

1. In the Watch List tab, click items to select them. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

2. Do one of the following:
   - In the Watch List tab at top right, click Analyze.
   - Right-click the selected items and click Analyze File.

The Watch List tab reports analysis results by updating its columns. For the first item selected, the Details tab provides individual status.

> To defragment a file or table

1. In the Watch List tab, click items to select them. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

2. Do one of the following:
   - In the Watch List tab at top right, click Defragment.
   - Right-click the selected items and click Defragment File.

The Watch List tab reports defragmentation results by updating its columns. For each item selected, the Details tab below provides individual status. For multiple selected items, the first one selected is shown.

> To cancel defragmentation

During defragmentation, click the Cancel Defragmentation icon and then click OK to confirm.

After cancellation, status for some items in the Watch List tab may be “Defrag: Completed,” with new statistics, while others display “Defrag: Canceled” and blank statistics. For a canceled item, the Details tab description gives the time taken until the moment of cancellation.

Successful cancellation means that fragmentation in the file or table is unchanged.

> To remove a file or table from the Watch List tab

1. In the Watch List tab, click items to select them. Use Ctrl-click to add one item at a time to the selection or shift-click to select a range. You can also use Ctrl-A to select all items.

2. Do one of the following:
   - In the Watch List tab at top right, click the Remove File icon.
   - Right-click the selected items and click Remove File.
To select columns to display
You can set every column in the Watch List tab to be shown or hidden.

1  In the Watch List tab at top right, click the Select Columns to Display icon.

2  By default, all choices are selected. Make any needed changes and click OK or click Cancel.

To hide or show details of analysis or defragmentation
You can toggle to hide or show the Details tab by clicking the Click to Hide Details icon.

Command Line Interface Defragmenter
The CLI utility dbdefrag provides the same functions as its GUI counterpart, except that there is no watch list. Defragmentation actions can be started, checked for status, or canceled from either interface.

The following table gives command line options.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbdefrag file</td>
<td>Start defragmentation on a file. Displays the number of records and keys processed. You can use Ctrl-C to cancel. In addition to path names, files can also be specified as a URI using the syntax brtv://user@host/dbname?parameters, as described under Database URIs.</td>
</tr>
<tr>
<td>dbdefrag-background file</td>
<td>Start defragmentation on a given file as a background process.</td>
</tr>
<tr>
<td>dbdefrag-cancel file</td>
<td>Cancel the currently running defragmentation for a file. Must be executed at a separate prompt unless the defragmentation is running in the background.</td>
</tr>
<tr>
<td>dbdefrag-status file</td>
<td>Show defragmentation status for the file. Displays the following for the most recently completed defragmentation:</td>
</tr>
<tr>
<td></td>
<td>• Status: Complete</td>
</tr>
<tr>
<td></td>
<td>• Started: yyyy-mm-dd hh:mm:ss</td>
</tr>
<tr>
<td></td>
<td>• Finished: yyyy-mm-dd hh:mm:ss</td>
</tr>
<tr>
<td></td>
<td>• Time taken: 0h:00m:00s</td>
</tr>
<tr>
<td></td>
<td>Displays the following for a defragmentation currently running:</td>
</tr>
<tr>
<td></td>
<td>• Defragmentation status: In Progress</td>
</tr>
<tr>
<td></td>
<td>• n out of n records processed (n%)</td>
</tr>
<tr>
<td></td>
<td>• n out of n keys processed (n%)</td>
</tr>
<tr>
<td></td>
<td>Displays the following for a canceled defragmentation:</td>
</tr>
<tr>
<td></td>
<td>• Status: Canceled</td>
</tr>
</tbody>
</table>
Monitoring Data File Fragmentation

The following example shows a series of steps to demonstrate use of `dbdefrag` on a large .mkd file.

1 Analyzing

C:\ProgramData\Actian\PSQL\Examples>dbdefrag -analyze mybtrievefile.mkd
Fragmented: 9%
Unused: 4%
Not in Order: 100%
File Size: 713 MB

2 Defragmenting

C:\ProgramData\Actian\PSQL\Examples>dbdefrag mybtrievefile.mkd
1125 out of 1195242 records processed (0%)
87021 out of 1195242 records processed (7%)
170910 out of 1195242 records processed (14%)
255393 out of 1195242 records processed (21%)
339805 out of 1195242 records processed (28%)
404202 out of 1195242 records processed (33%)
487928 out of 1195242 records processed (40%)
572585 out of 1195242 records processed (47%)
655802 out of 1195242 records processed (54%)
716472 out of 1195242 records processed (59%)
804006 out of 1195242 records processed (66%)
883729 out of 1195242 records processed (73%)
947475 out of 1195242 records processed (79%)
1032061 out of 1195242 records processed (86%)
1116015 out of 1195242 records processed (93%)
1185831 out of 1195242 records processed (99%)
1195242 out of 1195242 records processed (100%)
1 out of 2 keys processed (50%)
2 out of 2 keys processed (100%)
Defragmentation complete.
Time taken: 0h:24m:57s

3 Checking Status from Another Prompt or When Running in Background

C:\ProgramData\Actian\PSQL\Examples>dbdefrag -status mybtrievefile.mkd
Defragmentation status: In Progress
1053373 out of 1195242 records processed (88%)
0 out of 2 keys processed (0%)

Using dbdefrag

The following example shows a series of steps to demonstrate use of `dbdefrag` on a large .mkd file.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dbdefrag -analyze file</td>
<td>Show fragmentation statistics. Returns the following:</td>
</tr>
<tr>
<td></td>
<td>• File Size (MB)</td>
</tr>
<tr>
<td></td>
<td>• % Fragmented</td>
</tr>
<tr>
<td></td>
<td>• % Unused</td>
</tr>
<tr>
<td></td>
<td>• % Not in Order</td>
</tr>
<tr>
<td>dbdefrag -help</td>
<td>Display the listing of command options.</td>
</tr>
</tbody>
</table>
4 Checking Status After Defragmenting

C:\ProgramData\Actian\PSQL\Examples>dbdefrag -status mybtrievefile.mkd
  Status: Complete
  Started: 2014-08-08 11:08:28
  Finished: 2014-08-08 11:33:25
  Time taken: 0h:24m:57s

5 Analyzing Again

C:\ProgramData\Actian\PSQL\Examples>dbdefrag -analyze mybtrievefile.mkd
  Fragmented: 0%
  Unused: 0%
  Not in Order: 27%
  File Size: 682 MB
Monitoring Performance Counters

PSQL Server for Windows provides performance counters for use with the Windows Performance Monitor. (The PSQL performance counters are supported only on Windows Vista or later Windows operating systems.) The performance counters measure state or activity of the database engine, which allows you to analyze performance of your application. Windows Performance Monitor requests the current value of the performance counters at specified time intervals.

PSQL provides data only for display by the Performance Monitor and cannot modify the counter properties. Performance Monitor controls the following:

- The display of data is in three formats: line graph (default), histogram, and as a text report.
- The display fields are labeled Last, Average, Minimum, and Maximum.
- The scaling of values for display. PSQL provides a default scale for each counter. Performance Monitor allows you to change the scaling of individual counters. See To Change a Counter Scale.

The counter values reflect all calls into the database engine regardless of their source. That is, the MicroKernel Engine, Relational Engine, native Btrieve applications, utilities, and so forth, all contribute to the counter values. Counter values are collected for all files. Counters on a per-file basis are not currently supported.

Note that the use of performance counters is an advanced feature intended primarily for application developers and other technical staff. Refer to the Microsoft documentation for details about the Windows Performance Monitor and on the use of counters in general.

Registration During Installation

By default, the PSQL installation registers the PSQL performance counters with Performance Monitor. The counters are available for use after installation completes.

Note that response to customer needs may result in additional PSQL collector sets or counters being installed that are not discussed in this chapter. If so, refer to the description of the collector set or counter provided in Windows Performance Monitor. See Add Sets or Individual Counters To Monitor.

Data Collector Sets

A data collector set organizes multiple counters into a single component that can be used to review or log performance. PSQL provides the following data collector sets.

- PSQL MicroKernel Btrieve Operations
- PSQL MicroKernel Cache
- PSQL MicroKernel I/O
- PSQL MicroKernel Locks and Waits
- PSQL MicroKernel Transactions

PSQL MicroKernel Btrieve Operations

These counters are useful for characterizing the behavior of client applications in terms of the Btrieve API. The counters report the types of operations being processed by the database engine at a given point in time.

See also Btrieve API Operations in Btrieve API Guide.
The database engine uses a two-level memory cache system to increase performance of data operations. The two caches are called Level 1 (L1) Cache and Level 2 (L2) Cache.

The more frequently the engine must read a page from disk to complete a user request, the lower the performance. These counters can be used collectively to view how successfully the database engine avoids disk reads and to determine if any changes need to be made to the cache size settings.

The best performance occurs when all data is stored in the L1 Cache. Because of the sizes of data files, however, the L1 cache cannot always hold all of the data pages, so the database engine also uses a second level cache. Pages in the L2 Cache are stored in a compressed format, allowing for more pages to fit in memory. Consequently, it is lower performing than the database engine reading the pages from disk.

See also To calculate the ideal size of the database memory cache.

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Btrieve Close Operations per Second</td>
<td>The number of Btrieve Close operations per Second</td>
<td>To provide insight into the behavior of client applications. As a first step in troubleshooting issues, you may find it helpful to analyze your application behavior in terms of Btrieve operations.</td>
</tr>
<tr>
<td>Btrieve Get/Step Operations per Second</td>
<td>The number of Btrieve Get and Step operations per second.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>See Btrieve API Operations in Btrieve API Guide for the various Get and Step operations.</td>
<td></td>
</tr>
<tr>
<td>Btrieve Open Operations per Second</td>
<td>The number of Btrieve Open operations per Second</td>
<td></td>
</tr>
<tr>
<td>Btrieve Records Deleted per Second</td>
<td>The number of Btrieve records deleted per second</td>
<td></td>
</tr>
<tr>
<td>Btrieve Records Inserted per Second</td>
<td>The number of Btrieve records inserted per second</td>
<td></td>
</tr>
<tr>
<td>Btrieve Records Updated per Second</td>
<td>The number of Btrieve records updated per second</td>
<td></td>
</tr>
<tr>
<td>Change Operations per Second</td>
<td>The number of Brieve operations that modify the data files per second</td>
<td></td>
</tr>
<tr>
<td>Operations per Second</td>
<td>The number of Btrieve operations executed per second</td>
<td></td>
</tr>
<tr>
<td>Page Server Requests Per Second</td>
<td>The number of page server requests received per second</td>
<td></td>
</tr>
</tbody>
</table>

PSQL MicroKernel Cache

The PSQL MicroKernel Cache is a feature that allows the database engine to cache frequently accessed data in memory, improving performance. The cache is divided into two levels: Level 1 (L1) and Level 2 (L2). The L1 Cache is faster but smaller in size, while the L2 Cache is slower but can hold more data.

The more frequently the engine must access a page from disk to complete a user request, the lower the performance. These counters can be used collectively to view how successfully the database engine avoids disk reads and to determine if any changes need to be made to the cache size settings.

The best performance occurs when all data is stored in the L1 Cache. Because of the sizes of data files, however, the L1 cache cannot always hold all of the data pages, so the database engine also uses a second level cache. Pages in the L2 Cache are stored in a compressed format, allowing for more pages to fit in memory. Consequently, it is lower performing than the database engine reading the pages from disk.

See also To calculate the ideal size of the database memory cache.
### Table 50: Counters for MicroKernel Cache

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 1 Cache Dirty Percentage</td>
<td>The percentage of the Level 1 Cache in use that contains dirty pages</td>
<td>To help determine if heavily accessed pages are continuously being forced out of the cache, which may adversely affect performance. Dirty pages, ones with changes that have not been written to disk, may only reside in the L1 Cache. Under heavy write loads, the L1 Cache may predominately contain dirty pages. This forces pages out of the L1 Cache and into the L2 Cache, if configured, or out of the L1 Cache entirely. The database engine writes dirty pages to disk at scheduled intervals or when the L1 Cache gets close to full. Frequently writing pages to disk may also adversely affect performance. It may benefit performance to adjust the L1 Cache size so that the percentage of dirty pages is not always high. See also Cache Allocation Size.</td>
</tr>
<tr>
<td>Level 1 Cache Hits per Second</td>
<td>The number of Level 1 Cache hits per second</td>
<td>To help determine how successfully the database engine finds requested pages in L1 Cache. A higher rate of hits-to-misses indicates that the engine is finding pages in L1 Cache rather than needing to access the L2 Cache or physical storage.</td>
</tr>
<tr>
<td>Level 1 Cache Hit Ratio</td>
<td>The percentage of cache hits to total cache access for the L1 cache</td>
<td>The percentage displayed applies to the life of the database engine since the MicroKernel was last started.</td>
</tr>
<tr>
<td>Level 1 Cache Misses per Second</td>
<td>The number of Level 1 Cache misses per second</td>
<td></td>
</tr>
<tr>
<td>Level 1 Cache Usage</td>
<td>The percentage of Level 1 Cache currently in use</td>
<td>To aid in adjusting the size of the L1 Cache to fit your application(s). For example, applications that use small or predominately read-only data files may not fill up the L1 Cache as configured by default. The unused memory is not available to the operating system or to other applications. You can change the L1 Cache size to release the memory back to the operating system. Conversely, if you want to have an entire database in memory, you can monitor this value to know when the setting is as desired.</td>
</tr>
<tr>
<td>Level 2 Cache Hits per Second</td>
<td>The number of Level 2 Cache hits per second</td>
<td>To help determine how successfully the database engine finds requested pages in L2 Cache. A higher rate of hits-to-misses indicates that the engine is finding pages in L2 Cache rather than needing to access physical storage.</td>
</tr>
<tr>
<td>Level 2 Cache Hit Ratio</td>
<td>The percentage of cache hits to total cache access for the L2 cache</td>
<td>The percentage displayed applies to the life of the database engine since the MicroKernel was last started.</td>
</tr>
<tr>
<td>Level 2 Cache Misses per Second</td>
<td>The number of Level 2 Cache misses per second</td>
<td></td>
</tr>
</tbody>
</table>
Monitoring

The counters in this set are useful for understanding the interactions of the database engine and data read and written to physical storage. The pages reported by the counters are data file pages. These counters do not report data for pages in files used for archival logging or for transaction logging. See also Pages in PSQL Programmer’s Guide.

Table 50 Counters for MicroKernel Cache continued

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Level 2 Cache Raw Size</td>
<td>The current size of the Level 2 Cache in bytes</td>
<td>To help determine the size of the optional L2 Cache. The L2 Cache is one component of the setting for Max MicroKernel Memory Usage. That setting specifies the maximum proportion of total physical memory that the database engine is allowed to consume, which includes L1 Cache, L2 Cache, and all miscellaneous memory usage by the database engine. If the setting for Max MicroKernel Memory Usage is non-zero, the L2 Cache sizes itself to stay within the memory limit of the setting. The L2 Cache monitors memory consumption of the system and resizes itself as needed. The memory used by the L2 Cache may also be swapped out by the operating system.</td>
</tr>
<tr>
<td>Level 2 Cache Raw Usage</td>
<td>The amount of the Level 2 Cache currently in use in bytes</td>
<td>To show what percentage of the total system memory the L2 Cache is using.</td>
</tr>
<tr>
<td>Level 2 Cache Size Relative to Memory</td>
<td>Level 2 Cache size presented as a percentage of total system memory</td>
<td></td>
</tr>
<tr>
<td>Level 2 Cache Usage</td>
<td>The percentage of Level 2 Cache currently in use</td>
<td>To show what percentage of the Level 2 Cache is currently being used.</td>
</tr>
</tbody>
</table>

PSQL MicroKernel I/O

The counts in this set are useful for understanding the interactions of the database engine and data read and written to physical storage. The pages reported by the counters are data file pages. These counters do not report data for pages in files used for archival logging or for transaction logging. See also Pages in PSQL Programmer’s Guide.

Table 51 Counters for MicroKernel Input/Output

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pages Read per Second</td>
<td>The number of pages read from disk per second</td>
<td>To determine the interaction of the database engine and data read and written to physical storage.</td>
</tr>
<tr>
<td>Pages Written per Second</td>
<td>The number of pages written to disk per second</td>
<td></td>
</tr>
</tbody>
</table>

PSQL MicroKernel Locks and Waits

Client requests may be delayed by waiting for a resource to become available. These counters give insight into the types of database resources on which a client request may have to wait until the resource is available. As such, these counters may provide insight into the behavior of the database engine when multiple clients access it. A value close to or equal to the number of clients may indicate collisions for the same resources. Any corrective actions that can be done to alleviate these collisions may improve responsiveness.

The counters Waits on Page Buffers and Waits on Page Reads are waits on a global resource. All of the other counters in this grouping apply to multiple clients, but each client may be waiting on resources that differ from client to client.

See also Data Integrity and Supporting Multiple Clients, both in PSQL Programmer’s Guide.
Table 52  Counters for MicroKernel Locks and Waits

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Client Record Locks</td>
<td>The number of records explicitly locked by clients</td>
<td>To provide insight into the work load of client applications.</td>
</tr>
<tr>
<td>Waits on Active Reader Lock</td>
<td>The number of clients waiting on the Active Reader Lock. Multiple clients may hold the Active Reader Lock at the same time; however, the Active Reader Lock and the Active Writer Lock are exclusive. Consequently, a single client that holds the Active Reader Lock prevents any client from obtaining the Active Writer Lock. A single client that holds the Active Writer Lock prevents multiple clients from obtaining the Active Reader Lock. Each file has its own reader (and writer) lock. See also Waits on Active Writer Lock counter.</td>
<td></td>
</tr>
<tr>
<td>Waits on Active Writer Lock</td>
<td>The number of clients waiting on the Active Writer Lock. Only one client may hold the Active Writer Lock for a file at a time. Each file has its own writer (and reader) lock.</td>
<td>See also Waits on Active Reader Lock counter.</td>
</tr>
<tr>
<td>Waits on File Locks</td>
<td>The number of clients currently waiting on a file lock</td>
<td></td>
</tr>
<tr>
<td>Waits on Page Buffers</td>
<td>The number of clients waiting on a page buffer to become available. If a page is not available to service a request, the request blocks until the MicroKernel is able to make a page available.</td>
<td>To indicate whether or not the database engine has a page buffer available in the cache. Use this value along with the memory cache counters to decide if the caches are sized appropriately for the work load. Increasing the cache size will increase the total number of available pages, which can reduce the waits on page buffers. Three things may cause this value to spike when pages are not in cache: • a data file was recently opened • first time or infrequent access of a data page • the caches may be too small to contain all of the pages frequently accessed and modified. The spike for the first two items cannot be avoided because of accessing a file for the first time. The third item can be avoided by using a larger cache. If the caches are full and the cache misses are high, it is possible that the caches may be too small to contain all the pages frequently accessed and modified. See also Counters for MicroKernel Cache.</td>
</tr>
<tr>
<td>Waits on Page Locks</td>
<td>The number of clients currently waiting on page locks</td>
<td>To provide insight into the work load of client applications.</td>
</tr>
</tbody>
</table>
Monitoring

These counters are useful for understanding the behavior of client applications in terms of transactions. For example, a few long-lasting transactions that involve many changes cause a different behavior than many short-lived transactions.

See also Begin Transaction (19 or 1019), End Transaction (20), and Abort Transaction (21) in Btrieve API Guide, and MicroKernel Engine Fundamentals in PSQL Programmer's Guide.

### Using Windows Performance Monitor

This section provides some rudimentary instructions on using Windows Performance Monitor to get started with the PSQL performance counters. Refer to the Microsoft document for complete details on using Windows Performance Monitor.

The following steps assume that the PSQL performance counters have been registered with Windows Performance Monitor by the PSQL installation.

1. Display PSQL Data Collector Sets

   Start Windows Performance Monitor. The steps vary depending on the operating system, but generally the utility can be started from Control Panel --> Administrative Tools. You may also try the command "perfmon" in the Run window (Start --> Run).

<table>
<thead>
<tr>
<th>Table 52: Counters for MicroKernel Locks and Waits continued</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Waits on Page Reads</td>
<td>The number of clients waiting to read a page from disk. If a client is already in the process of reading the page, other clients must wait for the in-progress read to complete.</td>
<td>To help determine the number of clients trying to read the same page of the same file at the same time.</td>
</tr>
<tr>
<td>Waits on Record Locks</td>
<td>The number of clients currently waiting on record locks</td>
<td>To provide insight into the work load of client applications.</td>
</tr>
</tbody>
</table>

### PSQL MicroKernel Transactions

These counters are useful for understanding the behavior of client applications in terms of transactions. For example, a few long-lasting transactions that involve many changes cause a different behavior than many short-lived transactions.

Table 53: Counters for MicroKernel Transactions

<table>
<thead>
<tr>
<th>Counter</th>
<th>Description</th>
<th>Typical Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Transactions in Progress</td>
<td>The number of system transactions in progress. A system transaction is a special type of transaction that prepares data file changes then persists the changes to the file.</td>
<td>To help determine if system transactions are occurring too frequently or not often enough. The database engine writes changes to the data files during a system transaction. The frequency in which a system transaction occurs is determined by two server configuration parameters—Initiation Time Limit and Operation Bundle Limit—or triggered by a small amount of free space in the L1 Cache. In general, running the system transaction too frequently or not often enough adversely affects performance. Typically, you will notice that the number of page writes per second may increase, the number of Btrieve operations that modify records may decrease, and the number of clients waiting on the Active Writer Lock may increase. It may take experimentation to determine an ideal interval for a particular work load.</td>
</tr>
<tr>
<td>Transaction Commits per Second</td>
<td>The number of commits executed per second</td>
<td>To determine the number of application transaction commits. See also End Transaction (20) in Btrieve API Guide.</td>
</tr>
</tbody>
</table>
Monitoring Performance Counters

2. In the tree on the left, click “Performance Monitor,” then click the plus sign on the right.

3. In the “Available counters” group, scroll to the PSQL data collector sets.

Add Sets or Individual Counters To Monitor

1. Perform one of the following:
   a. To add an entire set, click the desired set in the “Available counters” group.
   b. To add individual counters, expand the desired set and click the desired counters.
To view a description of the counter, ensure that the **Show description** option is selected.

2 Click **Add**, then **OK**.

➢ **To Change a Counter Scale**
  1 Right-click the desired counter, then click **Properties**.

2 On the “Data” tab, click the “Scale” list, then click the desired value.
You may need to adjust the Scale to display two or more counters on the same graph that have vastly different ranges. The counter value is multiplied by the Scale value before the data is graphed. For example, assume that one counter outputs values of 53 and 99, and a second counter outputs 578 and 784. You may want to set the Scale for the first counter to 10 so that its output is 530 and 990. This lets you look at the data from both counters more comparably (530, 990, 578, and 784).

3 Click **OK**.

Note that the scale on the display changes from its original value ("1" in this example) to the new value ("10" in this example):

4 On the “Graph” tab, set the desired values for **Maximum** and **Minimum** of the “Vertical scale.”

You may want to change the vertical scale if what is being graphed is very small (or very large) so you can easily see when the values change. For example, if the counter values are always under 20, you may want to change the vertical scale to be **Maximum 20** and **Minimum 0**.

5 Click **OK**.
Monitoring License Usage

The PSQL products use different license models depending on the product. PSQL Server and Workgroup are licensed under a concurrent user count license. PSQL Vx Server is licensed under a capacity-based license.

<table>
<thead>
<tr>
<th>License Model</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Count License</td>
<td>The user count license model works well for traditional client/server applications in which many users or devices constantly add, update, and delete records from distinct individual desktops. Each product key specifies a licensed user count. A user count allows the specified number of concurrent connections to the PSQL database engine. Users are counted by network address.</td>
</tr>
<tr>
<td>Capacity-based License</td>
<td>A capacity-based model shifts the emphasis from how many users to how much work the database server processes. The model is based on capacity to accommodate license enforcement in service bureau, software-as-a-service, or other multiplexed environments. For example, each instance of PSQL Vx Server has capacity limits on both the number of sessions in use and the data in use.</td>
</tr>
</tbody>
</table>

Much of the monitoring pertaining to license usage involves user count, number of sessions, or data in use. For example, you may want to determine the current value, increase the current value, or better determine your capacity needs.

The four primary utilities used to monitor license usage are Monitor (see Monitoring Resource Usage), Capacity Usage Viewer, License Administrator, and Notification Viewer.

Capacity Usage Viewer

PCC provides Capacity Usage Viewer to monitor concurrent sessions and data usage for all database engines. This is especially useful when you are considering migrating from PSQL Server to PSQL Vx Server, because of the difference in the way those two editions are licensed.

Capacity Usage Viewer includes two graphs, one for the number of concurrent sessions and one for the amount of data. Each graph includes a usage level bar, a heavy horizontal line across the graph, to help you determine what volume of usage is normal and what volume is uncommon for your business. The Capacity Usage Viewer also displays peak usage statistics.

The graphs use the peak values that are recorded each day. For any day on which the engine is not used, they use a value of zero. They require a minimum of two days' data to be generated. Otherwise, Capacity Usage Viewer displays an error message.

This section contains the following topics:

- To access Capacity Usage Viewer
- Capacity Usage Viewer GUI
- Zooming

To access Capacity Usage Viewer

1. In PCC PSQL Explorer, right-click the engine you want to examine.
   - Its context menu opens.
2. Click Capacity Usage Viewer.
### Capacity Usage Viewer GUI

The following image shows the Capacity Usage Viewer. The table below the image describes its features. Click an area of the image for which you want more information.

**Figure 7  Capacity Usage Viewer User Interface**

![Capacity Usage Viewer User Interface](image)

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Bar</td>
<td>Identifies the engine you selected.</td>
</tr>
<tr>
<td>Statistical Indicators</td>
<td>Displays meaningful statistics and enables you to select what you want to display in the graphs. See <strong>Statistical Indicators</strong>.</td>
</tr>
<tr>
<td>Time Designation</td>
<td>Displays the starting and ending dates of the period for the data displayed in the graphs.</td>
</tr>
<tr>
<td>Session Usage Graph</td>
<td>Displays, graphically, the number of sessions that occurred concurrently during the selected time period.</td>
</tr>
<tr>
<td>Data Usage Graph</td>
<td>Displays, graphically, the amount of data used during the selected time period.</td>
</tr>
<tr>
<td>Usage Level Bars</td>
<td>Enables you to determine how often your usage exceeds a selected level. When selected, they appear across each graph at a default level that is 90% of peak usage. The number at the left of each usage level bar identifies its level (amount of data or number of sessions). You can move the usage level bar to whatever level you need, either by using the spin boxes or by dragging with the cursor. The two usage level bars are independent of each other.</td>
</tr>
<tr>
<td>Zoom Instructions</td>
<td>Describes the general procedures for zooming in and out of a graph. For detailed procedures, see <strong>Zooming</strong>.</td>
</tr>
<tr>
<td>Export Button</td>
<td>Enables you to export the data to a .CSV file, if you find it useful to save the data for additional analysis. The Export button opens a Browse for Folder dialog, where you can select a location for data storage.</td>
</tr>
</tbody>
</table>
Figure 8  Statistical Indicators

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
</tr>
</thead>
</table>
| Period                      | Enables you to select the time period for the data you want the graphs to display. When the window opens, the graphs display, by default, the data from the period that was selected when the window was last closed. You can select a different time period:  
  - All  
  - Last week  
  - Last 30 days  
  - Last 90 days  
  - Last 180 days  
  You can also select a time period by zooming the graphs. When you zoom a graph, the Period drop-down menu displays Custom as the selected time period. |
| Peak Statistics Group Box   | Contains fields that display statistics for maximum use of data and maximum number of concurrent sessions during the time period displayed in the graphs. |
| Peak Sessions               | Displays the greatest number of concurrent sessions that occurred during the time period displayed in the Session Usage graph.                  |
| Peak Sessions Date          | Displays the date on which the greatest number of concurrent sessions occurred. If that number of sessions occurred more on more than one day, the most recent date is displayed. |
| Peak Data (GB)              | Displays the maximum amount of data, in gigabytes, used at one time during the time period displayed in the Data Usage graph.                    |
| Peak Data Date              | Displays the date on which the maximum amount of data was used. If that amount of data was used on more than one day, the most recent date is displayed. |
| Show Usage Level Bars       | Displays or hides usage level bars across the graphs, depending on whether it is checked or not.                                           |
| Usage Statistics Group Box  | Contains spin boxes for moving the usage level bars up and down and fields that display the statistics that result from moving the usage level bars. |
| Sessions                    | Sets the value at which to position the usage level bar in the Session Usage graph.                                                           |
Zooming

If you need to view a particular time period besides the selections provided in the Period pull-down menu, you can select and zoom a segment of a graph. When you zoom one graph, the other zooms simultaneously. The two graphs are always set to the same period.

➢ To zoom a graph

1. Place the cursor at the start of the period you want to display.
2. With your left mouse button pressed, drag the cursor to the end of the period you want to display. In the graph, a rectangular black outline appears around the selected part of the graph. As you move the cursor, the rectangle enlarges accordingly.
3. Release the mouse button. The graph is rerendered to display the period you selected. The Period field displays the setting Custom.
4. Repeat the process to zoom further.
5. To zoom out, place the cursor anywhere in the graph and, with your left mouse button pressed, drag the cursor from right to left. The graph returns to its original period, that is, the period it displayed when the window opened.

License Administrator

License Administrator is the utility used to manage keys for licensing. It is fully documented in License Administration in PSQL User’s Guide. The following is a summary of the primary monitoring tasks that you can conduct with the utility, with links to PSQL User’s Guide.

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Days Exceeded</td>
<td>Displays the number of days on which the number of concurrent session in use was greater than the level at which the usage level bar is set.</td>
</tr>
<tr>
<td>Data (GB)</td>
<td>Sets the value at which to position the usage level bar in the Data Usage graph.</td>
</tr>
<tr>
<td>Days Exceeded</td>
<td>Displays the number of days on which the amount of data in use was greater than the level at which the usage level bar is set.</td>
</tr>
</tbody>
</table>
Table 54  Summary of License Usage Monitoring with License Administrator

<table>
<thead>
<tr>
<th>Item to Monitor</th>
<th>Using the Graphical User Interface</th>
<th>Using the Command Line Interface</th>
</tr>
</thead>
<tbody>
<tr>
<td>License information (All license information for an authorized key, such as the product, product key, status of a key, platform to which the license applies, license type, user count, session count, data in use, expiration date of a license, vendor software that installed the license, and application to which the license applies)</td>
<td>To Display License Information</td>
<td>To Display Information About a Key</td>
</tr>
<tr>
<td>User count (The number of allowed concurrent connections to the PSQL database engine permitted by a product key)</td>
<td>To Determine a Total User Count</td>
<td></td>
</tr>
<tr>
<td>Session count limit (The maximum permitted number of concurrent sessions as granted by a license agreement)</td>
<td>To Determine the Session Count Limit</td>
<td></td>
</tr>
<tr>
<td>Data in use limit (The maximum permitted total size of all concurrently open data files as granted by a license agreement)</td>
<td>To Determine the Data In Use Limit</td>
<td></td>
</tr>
<tr>
<td>Number of authorizations remaining for a key</td>
<td>To Display Remaining Authorizations</td>
<td>To Display Remaining Authorizations</td>
</tr>
</tbody>
</table>
Monitoring Database Access

Actian Corporation provides a companion product for PSQL called AuditMaster that is used to monitor access at the database level. AuditMaster allows you to monitor changes to mission-critical data for auditing purposes, such as the following:

- Who accessed a record or performed a change.
- What access or change occurred, when, and where originated.
- How the change was made.

AuditMaster provides a detailed audit trail and offers query and alert capabilities.

For product details, refer to AuditMaster User’s Guide included with AuditMaster. You can also find the guide on the PSQL website.
Monitoring

**Reviewing Message Logs**

PSQL provides various logging repositories for messages to assist you with troubleshooting. The logging falls into two broad categories:

- **All messages.** These messages include status, error, warning, and information messages. They can originate from any PSQL component, including the license administration components.
- **Licensing messages.** These messages alert you about licensing issues and provide troubleshooting information. They originate from license administration components.

The following table summarizes the repositories.

<table>
<thead>
<tr>
<th>Repository</th>
<th>Written To By</th>
</tr>
</thead>
<tbody>
<tr>
<td>Notification Viewer</td>
<td>License administration components</td>
</tr>
<tr>
<td>Operating System Event Log</td>
<td>License administration components (Windows)</td>
</tr>
<tr>
<td></td>
<td>All PSQL components (Linux)</td>
</tr>
<tr>
<td>PSQL Event Log (PVSW.LOG)</td>
<td>All PSQL components (Windows)</td>
</tr>
</tbody>
</table>

**Licensing Messages**

Several of the logging repositories emphasize licensing messages. If a key is determined to be invalid, the key changes state from active to failed validation. The database engine functions normally for a certain number of days so that you have ample time to correct the validation failures.

If you do not correct the failures before the number of days ends, the key changes state again to disabled. The key is no longer valid and the database engine cannot access data files.

Because you need to attend to a failed validation in a timely manner, the state change of the key is brought to your attention as soon as possible. For example, a message is logged to all of the message repositories. The most evident of these is PSQL Notification Viewer. (License Administrator also displays the state of keys, which you can recheck at any time by initiating a validation action. See License Administration in PSQL User's Guide.)
Change in State of Key

The following table explains the type of message returned based on the change in state of the key.

Table 56: Message Type Based on Change in State of Key

<table>
<thead>
<tr>
<th>Message Type</th>
<th>Change in State of Key</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>From</td>
<td>To</td>
</tr>
<tr>
<td>Warning</td>
<td>Active</td>
<td>Failed Validation</td>
</tr>
<tr>
<td>Warning</td>
<td>Disabled</td>
<td>Failed Validation</td>
</tr>
<tr>
<td>Error</td>
<td>Failed Validation</td>
<td>Disabled</td>
</tr>
<tr>
<td>Information</td>
<td>Failed Validation</td>
<td>Active</td>
</tr>
<tr>
<td>Information</td>
<td>Disabled</td>
<td>Active</td>
</tr>
</tbody>
</table>

Note that no messages are logged for keys with a state of “expired” (which applies only to temporary keys), or “inactive” (which applies to keys still registered on the machine from previous versions of PSQL).

Logging Frequency

The following table lists the frequency with which licensing messages are logged for particular actions.

Table 57: Logging Frequency of Messages By Initiating Action

<table>
<thead>
<tr>
<th>Initiating Action</th>
<th>Logging Frequency</th>
<th>Logging Repository¹</th>
</tr>
</thead>
</table>
| Key changes state as described in Table 56 | Immediately | • Notification Viewer  
• Operating System Event Log  
• PSQL Event Log |
| Key remains in failed validation state | Once a day reminder | • Notification Viewer  
• Operating System Event Log  
• PSQL Event Log |
| A validation action invoked programmatically through API call See To Display Remaining Authorizations in PSQL User’s Guide, PvValidateLicenses() in Distributed Tuning Interface Guide, and ValidateLicenses in Distributed Tuning Objects Guide. | Immediately | • Operating System Event Log  
• PSQL Event Log |
Monitoring

Notification Viewer

Notification Viewer is an application utility for displaying messages logged by the licensing components. The purpose of the utility is to inform you of noteworthy licensing messages (see Table 57) in a noticeable but unobtrusive manner.

By default, Notification Viewer is installed with PSQL Server, 32- and 64-bit, on Windows and Linux, and with PSQL Workgroup. Also by default on Windows platforms, Notification Viewer restarts when you restart Windows.

On Windows platforms, the executable is named notifyviewer.exe. The utility provides a single running instance for a user. An attempt to start Notification Viewer when it is already running brings the GUI to the front of the application displays.

On Linux distributions, the utility is a shell script named notifyviewer. Each time the shell script executes it starts another instance of Notification Viewer. If you restart Linux, you must restart Notification Viewer. The shell script is not automatically executed when Linux is restarted.

Command Line Options

You can specify how you want the utility to start with the following command line options.

<table>
<thead>
<tr>
<th>Option</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(no option)</td>
<td>If you start the utility without specifying an option, the GUI displays and a tray icon appears if the operating system supports a system tray.</td>
</tr>
<tr>
<td>-tray</td>
<td>Starts the utility with the GUI hidden and displays a tray icon. If the operating system does not support a system tray, the GUI displays (is not hidden).</td>
</tr>
</tbody>
</table>

Notification Viewer provides two interfaces: system tray icons and a graphical user interface (GUI).

System Tray Icons Interface

By default on Windows, Notification Viewer starts with the GUI hidden and displays its system tray icon. (On Linux, Notification Viewer starts as a GUI and displays its system tray icon if the Linux distribution supports a system tray.) After starting, the utility begins monitoring licensing messages.

If Notification Viewer detects unread messages, the tray icon visibly changes to indicate unread messages. See Tray Icons.

Notification Viewer also displays two types of tooltips. The mouse-over tooltip displays the number of important unread messages (if any), the total number of unread messages, or the name of the utility if all messages have been read. A balloon tooltip displays when Notification Viewer detects messages that need to be brought to your attention. On Windows, the balloon tooltip remains visible until you dismiss it.
it directly or perform a keyboard or mouse operation. On Linux, you must click the balloon tooltip to dismiss it.

**Popup Menu**

The popup menu for the tray icon contains two menu items: Open which opens the GUI, and Exit which closes the utility. Right-click the tray icon to display the menu.

**Tray Icons**

The following table explains the meaning of the tray icons.

<table>
<thead>
<tr>
<th>Icon</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Notification Viewer is running and monitoring licensing messages. This icon indicates a “normal” condition in which all messages have been read.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Notification Viewer contains unread messages. This icon remains visible until all unread messages are read. See Left panel.</td>
</tr>
</tbody>
</table>

**Graphical User Interface**

You can open Notification Viewer GUI by double-clicking the tray icon or by right-clicking the tray icon and clicking Open. By default on Linux, Notification Viewer starts as a GUI and displays its system tray icon. (If you want to change the startup behavior on Linux, pass the -tray option to the notifyviewer shell script. If the Linux distribution does not support a system tray, Notification Viewer displays the GUI but no system tray icon. In that case, start Notification Viewer by running the shell script.)

When the GUI is visible, unread messages are immediately added to the GUI. (In addition, the tray icon tooltip is shown and the icon changes to indicate unread messages.)

PSQL tracks which records in the notification file are read or unread for each user. That is, each user displaying the GUI sees all of the messages, but whether a particular message is read or unread varies by user.

If a system tray is not supported by an operating system, the Close command for Notification Viewer terminates the utility. If a system tray is supported, then closing hides the GUI and the tray icon becomes the only visual indication that the utility is still running.

**Toolbar and Panels**

The Notification Viewer GUI provides a toolbar and two main panels as shown in Figure 9.
Monitoring

Figure 9 Notification Viewer GUI

<table>
<thead>
<tr>
<th>GUI Element</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Toolbar</td>
<td>Provides two options:&lt;br&gt;• Show only unread messages (toggle on, toggle off)&lt;br&gt;• Mark all messages as read</td>
</tr>
<tr>
<td>Left panel</td>
<td>Contains a scrollable list of messages. They are sorted with the latest messages on top and arranged in three groups: &quot;Today,&quot; &quot;Previous 7 days,&quot; and &quot;Older.&quot; You can expand and collapse each group. Each message shows an image for the type of information, a caption string, the message date and a brief description. All text is in a bold typeface if the message is unread. To &quot;read&quot; a message: click (select) the message. Once selected, the typeface for the message changes to a regular typeface.</td>
</tr>
<tr>
<td>Right panel</td>
<td>Shows the message details, which provide a full description of the message, and when applicable, suggestions to resolve any issues.</td>
</tr>
</tbody>
</table>

**Operating System Event Log**

The PSQL license administration components write messages to the operating system event log on Windows platforms. All PSQL components write messages to the operating system event log on Linux distributions.

**Windows Platforms Event Logs**

Windows operating systems provide a method to log events categorized as “Application,” “Security,” or “System.” PSQL logs licensing messages only to the Windows Application Event Log. The messages are a subset of the licensing messages written to PVSW.LOG.

Any licensing message categorized as an “error” or “warning” is logged. This includes messages that result from a change in the state of a key as well as other warning and error messages (see Table 57). In addition, certain information messages are logged, such as those listed in Table 56.

**Viewing Event Logs**

Windows operating systems provide a graphical user interface utility called Event Viewer to view and manipulate event logs. The utility can be accessed through the Windows PC Settings or Control Panel or by executing the command eventvwr.msc from a command interface.

PSQL displays the following for an event.

- **Date and Time** - date and time the event occurred
- **Source** - Pervasive SQL
- Category - Product Keys
- Type/Level - level of the event: Information, Warning, or Error
- Event ID - 1000
- User - N/A
- Computer - name of the computer

In addition, the Keyword column displays “Classic” and the Log column displays “Application.” Event Viewer allows you to display additional columns, but PSQL provides no data for them.

**Linux Distributions Event Logs**

On Linux distributions, all PSQL components write messages to the standard Linux logging system, syslog. By default, syslog writes to the /var/log/messages directory. Optionally, for SQL Connection Manager only, you can also log messages to the event.log file.

**Event.log File and Bti.ini**

Bti.ini is a PSQL configuration file used on Linux distributions. By default, the file is located in the /usr/local/psql/etc directory.

The file lets you configure settings for the SQL Connection Manager (the SQL Manager section in the .ini file). One of the settings, LogEvent, determines the type of event messages logged to the event.log file. By default, event.log is located in the /usr/local/psql/bin directory.

<table>
<thead>
<tr>
<th>Bti.ini Parameters for SQLManager Section</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MgrPort</td>
<td>Sets the port number used by the SQL Connection Manager. The default is 1583.</td>
</tr>
<tr>
<td>MgrUseTransport</td>
<td>Sets the type of protocol used by the SQL Connection Manager. This must be set to TCP.</td>
</tr>
</tbody>
</table>
| LogEvent=msg_type | Specifies one of the following values for msg_type to indicate the type of messages logged to event.log (the default is 1):  
  • 0 – no logging  
  • 1 – errors only  
  • 2 – errors plus warnings  
  • 3 – errors plus warnings plus information messages  
  • 4 – errors plus warnings plus information messages plus connect.log |
| InstallDirectory=/usr/local/psql | Activates the connection log: /usr/local/psql/connect.log |

**PSQL Event Log (PVSW.LOG)**

On Windows platforms, all PSQL components write status, error, warning, and information messages to the PSQL event log. (On Linux distributions, PSQL does not use an exclusive event log. Instead, all PSQL components write messages to the standard Linux logging system, syslog. See Linux Distributions Event Logs.)

The PSQL event log is called PVSW.LOG and, by default, is located in the application_data_directory\PSQL\logs directory. All PSQL components on Windows platforms write to this log file. If two or more applications using the PSQL database engine are running on the same machine, they share PVSW.LOG.
**PVSW.LOG Fields**

PVSW.LOG consists of text messages in a format described in Table 58.

<table>
<thead>
<tr>
<th>Field</th>
<th>Contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date</td>
<td>Automatic date-stamp in <em>mm/dd/yyyy</em> format.</td>
</tr>
<tr>
<td>Time</td>
<td>Automatic time-stamp in <em>hh:mm:ss</em> format. Also indicates AM or PM.</td>
</tr>
<tr>
<td>Component</td>
<td>File name of component returning the error (prefix only, no extension).</td>
</tr>
<tr>
<td>Process</td>
<td>Instance ID of the component, which is the process ID of the component.</td>
</tr>
<tr>
<td>Process Name</td>
<td>Path and name of the component, truncated to the last 15 characters.</td>
</tr>
<tr>
<td>Computer Name</td>
<td>Name assigned to the machine hosting the process, truncated to the first 15 characters.</td>
</tr>
<tr>
<td>Type</td>
<td>A single character: I for Information, W for Warning, or E for Error.</td>
</tr>
<tr>
<td>Message</td>
<td>The message text which may be either a string retrieved from a resource associated with the calling</td>
</tr>
<tr>
<td></td>
<td>component or a text string passed directly from the calling component.</td>
</tr>
<tr>
<td></td>
<td>Some message text may contain numeric values, which may be in decimal or hexadecimal format.</td>
</tr>
<tr>
<td></td>
<td>The characters &quot;0x&quot; precede any hexadecimal values to distinguish them from decimal values.</td>
</tr>
<tr>
<td></td>
<td>Some message text may also contain information specific to an OEM application, such as a link to</td>
</tr>
<tr>
<td></td>
<td>a vendor's Web site and troubleshooting information.</td>
</tr>
</tbody>
</table>

An entry may be followed by binary data in standard hexadecimal format. There is no limit to the length of the binary data.

**PVSW.LOG Example Entry**

The following shows an example of the type of data contained in PVSW.LOG.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Component</th>
<th>Process</th>
<th>Process Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>5/10/2011</td>
<td>9:53:06 AM</td>
<td>LicenseMgr</td>
<td>9048</td>
<td>NTDBSMGR.EXE</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Computer Name</th>
<th>Type</th>
<th>Message</th>
</tr>
</thead>
<tbody>
<tr>
<td>USRegion2Svr</td>
<td>W</td>
<td>License failed validation. Remaining Days: 14</td>
</tr>
</tbody>
</table>
Receiving Email Notification of Messages

PSQL does not include email notification of event messages because products from other vendors that provide such functionality are readily available. This section identifies some of those products. Also discussed is the event log content to monitor for events pertaining to PSQL licensing and product keys.

Products That Provide Email Notification of Monitored Events

The following table is a partial list of products that can provide email notification of events in the operating system event log. The products are listed alphabetically. Actian Corporation does not endorse one product over another. Since the products listed are from other vendors, the products and the discussion about them may vary from what is presented here.

Note that such products typically require either an agent to be installed or a remote access mechanism like Windows Management Instrumentation (WMI) or secure shell (SSH) to be enabled.

Table 59  Products That Provide Email Notification of Monitored Events

<table>
<thead>
<tr>
<th>Product</th>
<th>Cost</th>
<th>Platform</th>
<th>Discussion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hyperic</td>
<td>Contact Hyperic</td>
<td>Windows</td>
<td>Requires installing an agent on the monitored machine and opening a firewall port for the agent.</td>
</tr>
<tr>
<td><a href="http://www.hyperic.com">www.hyperic.com</a></td>
<td></td>
<td>Linux and Mac OS X</td>
<td>Requires enabling SSH on the monitored machine and opening a firewall port for SSH.</td>
</tr>
<tr>
<td>Nagios</td>
<td>Free</td>
<td>Windows and Linux</td>
<td>Requires installing an agent on the monitored machine and opening a firewall port for the agent.</td>
</tr>
<tr>
<td><a href="http://www.nagios.org/">www.nagios.org/</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Spiceworks</td>
<td>Free</td>
<td>Windows</td>
<td>Requires enabling remote WMI on the monitored machine and opening a firewall port for remote WMI.</td>
</tr>
<tr>
<td><a href="http://www.spiceworks.com">www.spiceworks.com</a></td>
<td></td>
<td>Linux</td>
<td>Requires enabling SSH on the monitored machine and opening a firewall port for SSH.</td>
</tr>
<tr>
<td>System Center Configuration Manager (SCCM)</td>
<td>Contact Microsoft</td>
<td>Windows</td>
<td>Requires either an agent or enabling remote WMI on the monitored machine. The firewall is automatically adjusted when WMI is enabled. If an agent is used, also requires opening a firewall port for the agent. This product is available only for Windows operating systems.</td>
</tr>
<tr>
<td><a href="http://www.microsoft.com">www.microsoft.com</a></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ZenOSS</td>
<td>Contact ZenOSS</td>
<td>Windows</td>
<td>Requires installing an agent on the monitored machine and opening a firewall port for the agent.</td>
</tr>
<tr>
<td><a href="http://www.zenoss.com">www.zenoss.com</a></td>
<td></td>
<td>Linux and Mac OS X</td>
<td>Requires enabling SSH on the monitored machine and opening a firewall port for SSH.</td>
</tr>
</tbody>
</table>

Event Log Content to Monitor for License Event Messages

All of the products in Table 59 have the ability to identify content in the operating system event log. The identification methods differ between Windows platforms and Linux distributions. The intent of this section is not to discuss the specific monitoring methods of each product. Refer to the vendor documentation for product details.
**Windows Platforms**

The following table identifies the properties in the operating system event log specific to PSQL licensing. Configure your monitoring product to send email notifications based on the values of those properties.

<table>
<thead>
<tr>
<th>Operating System Event Log Property</th>
<th>Value to Monitor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source</td>
<td>Pervasive SQL</td>
</tr>
<tr>
<td>Category</td>
<td>Product Keys</td>
</tr>
<tr>
<td>Type</td>
<td>Warning, Error</td>
</tr>
</tbody>
</table>

**Examples**

Suppose that you want to receive email notifications for all PSQL licensing event messages that are warnings or errors. Configure your product to monitor the following conditions.

- Source=“Pervasive SQL”
- Category=“Product Keys”
- Type=“Warning” | Type= “Error”

**Linux Distributions**

The Linux syslog does not contain the same properties as the Windows operating system event log. For Linux, you must configure the notification product to identify events and strings in the syslog. Based on events and string comparisons, the product can then initiate the desired action, such as send an email notification.

Note that all messages—whether licensing related or not—written to the syslog by the PSQL database engine contain the string “mkded.” The following list identifies some of the strings in the syslog specific to PSQL licensing. One way to monitor the syslog is to configure a string comparison based on content that contains “mkded” and one of the following strings:

- disabled
- failed validation
- for key
- validation of key
- capacity for session count
- capacity for data in use
- user count
Testing Btrieve Operations

How to Perform Btrieve Operations with the Function Executor Utility

This chapter discusses the following topics:

- Function Executor Concepts
- Function Executor Graphical User Interface
- Function Executor Tasks
Function Executor Concepts

This section contains the following topics:

- Overview
- What Function Executor Can Do
- Function Executor Features
- Automatic Mode in Function Executor
- Where to Learn More

Overview

Function Executor runs on Windows. With this interactive utility, you can learn how Btrieve operations work. (This chapter refers to operations for the MicroKernel Engine as “Btrieve operations.”)

By allowing execution of Btrieve operations one at a time, Function Executor enables application developers to simulate the operations of a Btrieve application. This simulation can isolate the database calls from the rest of your application, which can help in testing and debugging your program.

Function Executor is primarily a tool for application developers. This chapter assumes that you have a basic knowledge of Btrieve operations. For more information about Btrieve operations, refer to the Btrieve API Guide that is available in the Developer Reference.

What Function Executor Can Do

- Perform Btrieve operations while monitoring the contents of memory structures.
- Allow you to capture a series of Btrieve operations and save them as a history file for later playback.
- Display the Btrieve version for clients, and local and remote engines.
- Display the Btrieve characteristics of data files and allow you to save those characteristics as a template (description file) or create a new file based on those characteristics. See File Statistics for more information.

Function Executor Features

Function Executor features include the following:

- Editor Status Bar
- Statistics
- Get and GetExt
- Transaction Toolbar
- Login Dialog
- History Log
- Viewing as Any Data Type
Editor Status Bar

The status bar contains the following elements:

The last/current status code is shown in the editor window’s status bar at the bottom of the window for the open file, and appears red if the last status was not zero.

Placing the mouse cursor over the status code shows a description of the status and what operation caused it. You can also click the display in red in order to display the full help for the status code. See To get help for a status code for more information.

When your cursor is in the input area on the main window of Function Executor, the status bar displays the offset within the buffer: the hex, the decimal, and the ASCII value of the byte you are presently on.

The status bar also indicates how many operations have been performed when there are multiple items in the queue. Normally this displays 1 of 1, but if you are executing multiple operations, it will display the count as the operations are executed.

The status bar also displays when you are in a transaction.

Statistics

Clicking the File Statistics icon displays a dialog box listing statistics for the currently open file. You can print these statistics to a text file or save them to a description file usable by BUTIL -CREATE. You may also create a new blank file with the same characteristics.

### Statistics

<table>
<thead>
<tr>
<th>File Size</th>
<th>1,877,283</th>
<th>Page Size (Bytes)</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Version</td>
<td>9.5</td>
<td>Number of Records</td>
<td>1503</td>
</tr>
<tr>
<td>Record Length</td>
<td>4.5</td>
<td>Unused Prelocated Pages</td>
<td>0</td>
</tr>
<tr>
<td>Number of Keys/Segments</td>
<td>559</td>
<td>Unused Linked Duplicate Pointers</td>
<td>0</td>
</tr>
<tr>
<td>Open Mode</td>
<td>Normal</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The following table shows the key information for the open file:

<table>
<thead>
<tr>
<th>Pos</th>
<th>Size</th>
<th>Type</th>
<th>Free Space Threshold</th>
<th>Key Only</th>
<th>Index Balancing</th>
<th>Page Partialization</th>
<th>Unused Entries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3125</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>1400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3125</td>
</tr>
<tr>
<td>3</td>
<td>4</td>
<td>1400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3125</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1400</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3125</td>
</tr>
</tbody>
</table>

### Example

**Operation B_GET_NEXT (6) returned status 8**

The current positioning is invalid.

**Pos = 0 Hex(38) Dec(56) Asc(8)**

1 of 1
Testing Btrieve Operations

Get and GetExt

From the Get menu, you can retrieve the First, Next, Previous, and Last records in the table. The GetExt menu includes the Goto Percent, Get Position, and Find Percent commands.

The Get and GetExt commands are available from both the menu bar and toolbar. The toolbar offers Step (Physical) and Get (Logical), allowing you to move either through the natural order of the file (physical) or in a specific order (logical).

Goto Percent allows you to choose whether to jump to a point within the physical layout of the file, or down any key path, limited to the keys defined in the file. You can also set lock biases using the option buttons in the Locks group box.

Find Percentage is the opposite of Goto Percent. It tells you how far into the data you are, depending on whether you are stepping through the file logically or physically.

Transaction Toolbar

The Transaction toolbar lets you start, stop, and abort transactions. You can set all aspects of the Transaction API through this toolbar, and the operation is executed immediately. The Transaction status also appears on the main window status bar, since it affects all open files for the client ID.

Login Dialog

The Login dialog box allows you to perform the Btrieve login operation via a GUI interface. See the following topics for more information about the Login dialog box:

- Login and Logout
- PSQL Databases
- PSQL Security
History Log

When you perform operations using the Function Executor utility, they are recorded in a History log. You can use this log to perform the operations contained therein, or save the history as a file that you can later reload and step through.

![Image of Function Executor utility]

See the following topics for more information about the History log:
- History
- History Tasks

Viewing as Any Data Type

When a file is open, you can right-click any position in the buffer and select **Show As**. A dialog box appears in which you can view the bytes at the chosen buffer position as any data type.

![Image of View as Any Data Type]

Table 60 lists controls available Function Executor.

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Repeat</td>
<td>Allows you to repeat commands.</td>
</tr>
<tr>
<td>Create</td>
<td>Allows you to create a new file.</td>
</tr>
<tr>
<td>File Statistics</td>
<td>Gives information from the BSTAT function. You can print these statistics.</td>
</tr>
<tr>
<td>MDI</td>
<td>The Multiple Document Interface permits the opening of multiple files.</td>
</tr>
</tbody>
</table>
Testing Btrieve Operations

Table 60  Function Executor Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reset</td>
<td>Reset Client ID</td>
</tr>
<tr>
<td>Stop</td>
<td>Btrieve Stop</td>
</tr>
<tr>
<td>Version</td>
<td>Btrieve Version</td>
</tr>
</tbody>
</table>

**Automatic Mode in Function Executor**

For each open file (see Application Window), you have the option of performing Btrieve operations with or without the assistance of Function Executor. You do not need to make any configuration changes to use one method or the other. Whether you use automatic mode or manual mode depends on which GUI controls you use.

**Note**  Selections from the menus (see Application Window) also are part of the automatic mode.

When you click a button in the automatic mode area, the following assistance is provided by the utility:

- Data buffers and data lengths are set automatically to prevent Status code 22.
- Information is requested appropriate to the operation.

**Where to Learn More**

Function Executor is a valuable tool for program developers, but it assumes you have a working knowledge of Btrieve fundamentals. Refer to the following topics to understand all the features of this utility:

- The chapter **MicroKernel Engine Fundamentals** in PSQL Programmer’s Guide. That guide is part of the PSQL Developer Reference.
- The chapter **Btrieve API Operations** in Btrieve API Guide. That guide is part of the PSQL Developer Reference.
- The chapter **PSQL Security**.
Function Executor Graphical User Interface

This section describes the objects on Function Executor graphical user interface (GUI).

- Application Window
- Main Window
- Login and Logout
- Open File Dialog
- Create a Btrieve File
- Create a File Dialog GUI Reference (Advanced)
- Transactions
- File Statistics
- History

**Application Window**

The table below the following image explains the window components. Click an area of the image for which you want more information.

![Function Executor Application Window](image)

Table 61  Function Executor Application Window

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
</table>
| **File menu** | Allows you to perform the following commands and Btrieve operations:  
• Login and Logout  
• Open and Close  
• New  
• Print Setup  
• Set Owner Name and Clear Owner Name  
• Start Continuous Operations or End Continuous Operations  
• Reset, Stop, and Exit | Performing Operations Tasks  
Opening a File Tasks |
| **Get menu** | Allows you to perform the following Btrieve operations:  
• Get First and Get Next  
• Get Previous and Get Last  
• Get Greater Than and Get Greater or Equal  
• Get Less and Get Less or Equal | Performing Operations Tasks |
Testing Btrieve Operations

Table 61  Function Executor Application Window  continued

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>GetExt menu</td>
<td>Allows you to perform the following Btrieve operations:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Goto Percent</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td></td>
<td>• Get Position</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Find Percent</td>
<td></td>
</tr>
<tr>
<td>Step menu</td>
<td>Allows you to perform the following Btrieve operations:</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td></td>
<td>• Step First and Step Next</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Step Previous and Step Last</td>
<td></td>
</tr>
<tr>
<td>Updates menu</td>
<td>Allows you to perform the following Btrieve operations:</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td></td>
<td>• Insert, Update, and Delete</td>
<td></td>
</tr>
<tr>
<td></td>
<td>You can also release locks using this menu.</td>
<td></td>
</tr>
<tr>
<td>View menu</td>
<td>Allows you to display GUI elements:</td>
<td>History Tasks</td>
</tr>
<tr>
<td></td>
<td>• Toolbars (main and transactions)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• History window</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• File statistics window</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Engine version using the Btrieve Version operations</td>
<td></td>
</tr>
<tr>
<td>Tools menu</td>
<td>Allows you to perform the following Btrieve operations:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Get Directory and Set Directory</td>
<td></td>
</tr>
<tr>
<td>Window menu</td>
<td>Allows you to perform windowing operations:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Cascade windows and Tile windows</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Select from a list of open windows</td>
<td></td>
</tr>
<tr>
<td>Help menu</td>
<td>Allows you to select from a list of help resources for this utility.</td>
<td>To get help for a status code</td>
</tr>
<tr>
<td>Open</td>
<td>Displays a dialog box from which you select a Btrieve file to open.</td>
<td>Opening a File Tasks</td>
</tr>
<tr>
<td>Create</td>
<td>Displays a dialog box with which you can create a new Btrieve file.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>Reset</td>
<td>Resets the current client connection (Btrieve operation 28) and closes all</td>
<td></td>
</tr>
<tr>
<td></td>
<td>files opened with that client ID.</td>
<td></td>
</tr>
<tr>
<td>Stop</td>
<td>Terminates transactional services (Btrieve operation 25) and closes all open</td>
<td></td>
</tr>
<tr>
<td></td>
<td>files.</td>
<td></td>
</tr>
<tr>
<td>Statistics</td>
<td>Displays a dialog box listing the currently opened file’s statistics. You</td>
<td></td>
</tr>
<tr>
<td></td>
<td>can print these statistics to a text file or save them to a description file</td>
<td></td>
</tr>
<tr>
<td></td>
<td>usable by BUTIL -CREATE.</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>Displays information about the version of PSQL (using Btrieve operation 26)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>that you are running. If no file is open, you will see information about the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>requester DLLs and any local MicroKernel engine. If you open a file on a</td>
<td></td>
</tr>
<tr>
<td></td>
<td>remote server, you will see information about the server and requester DLLs.</td>
<td></td>
</tr>
</tbody>
</table>
Table 61  Function Executor Application Window  continued

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status Codes Help</td>
<td>If no file is open, displays Status Codes help file. If file is open and a status code is active, displays help for that particular status code.</td>
<td></td>
</tr>
<tr>
<td>Show help</td>
<td>Toggles whether a pop-up dialog box displays when a non-zero status code is received.</td>
<td></td>
</tr>
<tr>
<td>Show history</td>
<td>Toggles whether the History window is displayed.</td>
<td>To display the History window History History Log</td>
</tr>
<tr>
<td>Operation count</td>
<td>Indicates the current operation number. This is used when you set the Repeat to a value greater than one.</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Transaction state</td>
<td>Indicates the current state of any transactions. Can be: • Blank, if no transaction is in effect • Concurrent • Exclusive</td>
<td>Performing Operations Tasks Transactions</td>
</tr>
</tbody>
</table>

**Main Window**

A main window displays for every open file. Click an area of the image for which you want more information.
## Testing Btrieve Operations

### Table 62  Function Executor Main Window for an Open File

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title Bar</td>
<td>Lists the full path of the open data file.</td>
<td>To open a data file with Function Executor</td>
</tr>
<tr>
<td>Data Buffer</td>
<td>Specifies a data value. For read and write operations, the Data Buffer contains records. For other operations, the Data Buffer contains file specifications, filtering conditions, and other information the database engine needs for processing the operation. This control corresponds with the Data Buffer parameter.</td>
<td></td>
</tr>
<tr>
<td>Key Buffer</td>
<td>Specify the path for the data file for which you want to perform a Btrieve operation.</td>
<td></td>
</tr>
<tr>
<td>Step vs. Get</td>
<td>Toggles between Step and Get operations</td>
<td></td>
</tr>
<tr>
<td>Get/Step First</td>
<td>Performs the Get or Step Next operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get/Step Prev</td>
<td>Performs the Get or Step Previous operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get/Step Next</td>
<td>Performs the Get or Step Next operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get/Step Last</td>
<td>Performs the Get or Step Last operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get Equal</td>
<td>Performs the Get Equal operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get Less Than</td>
<td>Performs the Get Less Than operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get Greater Than</td>
<td>Performs the Get Greater Than operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get Less or Equal Than</td>
<td>Performs the Get Less or Equal Than operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get Greater or Equal Than</td>
<td>Performs the Get Greater or Equal Than Operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Get/Find Percent</td>
<td>Performs the Get or Find Percent operations</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Insert</td>
<td>Performs the Insert operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Update</td>
<td>Performs the Update operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Delete</td>
<td>Performs the Delete operation</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Cancel</td>
<td>Cancels the recent changes</td>
<td></td>
</tr>
<tr>
<td>Unlock</td>
<td>Releases any locks.</td>
<td></td>
</tr>
<tr>
<td>Set or Goto Bookmark</td>
<td>Sets or positions at a previously defined bookmark.</td>
<td></td>
</tr>
<tr>
<td>Key Num</td>
<td>For most Get operations, specifies a key number, or index path, to follow for the current operation. For other operations, specifies such information as file open mode, encryption, or logical disk drive. This control corresponds with the Key Number parameter.</td>
<td></td>
</tr>
<tr>
<td>Key Only</td>
<td>Specifies to get key only, not data.</td>
<td></td>
</tr>
<tr>
<td>Repeat</td>
<td>Repeats the operation the number of times you specify.</td>
<td></td>
</tr>
</tbody>
</table>
Login and Logout

The Login and Logout dialog boxes allow you to perform the Btrieve login and logout operations, respectively, for the database engine. Click an area of the image for which you want more information.

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Locks</td>
<td>Specifies the locking behavior you want in for the current operation.</td>
<td></td>
</tr>
<tr>
<td>Op Code</td>
<td>Specifies the current operation code plus its bias (if any). The default is 0. If you are familiar with Btrieve operation codes, you can enter the desired code. Otherwise, use the List box to specify an operation. This control corresponds with the Operation Code parameter.</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Execute button</td>
<td>Performs the currently specified operation.</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Operations list</td>
<td>Lists all Btrieve operations and their codes. The default is Open (0). You can move quickly through the list by entering the first letter of the operation you want to perform.</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>DataLen</td>
<td>Specifies the length (in bytes) of the Data Buffer. The default is 1024. For every operation that requires a data buffer, you must specify a buffer length. On many operations, the database engine returns a value to the Data Length. Generally, you should always specify a Data Length before you execute an operation. This control corresponds with the Data Buffer Length parameter.</td>
<td>Performing Operations Tasks</td>
</tr>
<tr>
<td>Status Code Indicator</td>
<td>Displays a numeric status code returned by the database engine and a brief message explaining the result of a Btrieve operation. For detailed information about these status codes and messages, refer to the Status Codes and Messages manual.</td>
<td>To get help for a status code</td>
</tr>
<tr>
<td>Continuous Operations Indicator</td>
<td>Displays the following on the bottom row of the file window if the file is in Continuous Operations mode (operation 42).</td>
<td>Using Continuous Operations</td>
</tr>
</tbody>
</table>
Testing Btrieve Operations

Figure 11 Login Dialog

Figure 12 Logout Dialog

Table 63 Function Executor Login and Logout Dialogs

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Server</td>
<td>Specifies the server where the database resides that you wish to log in to or log out from. Note that a server name is required to access a database on a Linux operating system. See also Database URIs in PSQL Programmer's Guide.</td>
</tr>
<tr>
<td>Database</td>
<td>Specifies the database on the server to which you want to authenticate.</td>
</tr>
<tr>
<td>User Name</td>
<td>The user name you want to authenticate against the database.</td>
</tr>
<tr>
<td>Password</td>
<td>The password for the user name.</td>
</tr>
<tr>
<td>Client ID</td>
<td>If you want this login or logout to apply to a specific client ID, click Use and specify the range. Otherwise, leave as is. See Client ID in Btrieve API Guide.</td>
</tr>
<tr>
<td>URI String</td>
<td>As you enter information in the forms, the URI resulting from your selections is shown in this area.</td>
</tr>
</tbody>
</table>

Open File Dialog

This dialog box allows you to open a file. Click an area of the image for which you want more information.
Table 64  Function Executor Open File Dialog

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>Specifies the location and name of the file that you want to open.</td>
<td>Opening a File Tasks</td>
</tr>
<tr>
<td>Owner name</td>
<td>Specifies a password to associate with the Btrieve file. If specified, the owner name is required to gain access to the Btrieve file. Owner name has no relation to any system user name. You can think of an owner name as a file password. A “short” owner name can be up to 8 bytes. A “long” owner name can be up to 24 bytes. Note, however, that once a long owner name is specified, the data file cannot be read by a database engine prior to PSQL v10.10. Also, a data file with a long owner name cannot be rebuilt to a file format prior to 9.5 unless the owner name is first removed. An owner name, long or short, with less than the maximum allowed bytes is padded with spaces to the maximum length (8 or 24 bytes).</td>
<td>Opening a File Tasks See also the chapter PSQL Security.</td>
</tr>
<tr>
<td>Mode</td>
<td>Specifies the state of the file when it is opened. Based on the state, the database engine knows certain conditions that apply to the opened file. For example, a condition could be that the file can be read but not updated (read-only).</td>
<td>Opening a File Tasks See Open Modes in Btrieve API Guide in the Developer Reference for an explanation of the modes.</td>
</tr>
<tr>
<td>Client ID</td>
<td>If you want this login to apply to a specific client ID, click Use and specify the range. Otherwise, leave as is.</td>
<td>Opening a File Tasks</td>
</tr>
<tr>
<td>Sharing</td>
<td>The database engine ignores the “Sharing” options. The “Sharing” options applied only to a legacy version of the engine, Btrieve 6.15.</td>
<td>Opening a File Tasks</td>
</tr>
</tbody>
</table>

Create a Btrieve File

This dialog box allows you to create a Btrieve file. Click an area of the image for which you want more information.
Testing Btrieve Operations

Table 65  Function Executor Create File Dialog

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Filename</td>
<td>Specifies the location and name of the file that you want to create.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>Owner name</td>
<td>Specifies a password to associate with the Btrieve file. If specified, the owner name is required to gain access to the Btrieve file. Owner name has no relation to any system user name. You can think of an owner name as a file password. A “short” owner name can be up to 8 bytes. A “long” owner name can be up to 24 bytes. Note, however, that once a long owner name is specified, the data file cannot be read by a database engine prior to PSQL v10.10. Also, a data file with a long owner name cannot be rebuilt to a file format prior to 9.5 unless the owner name is first removed. An owner name, long or short, with less than the maximum allowed bytes is padded with spaces to the maximum length (8 or 24 bytes).</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
</tbody>
</table>

| Mode       | Specifies the state of the file when it is opened. Based on the state, the database engine knows certain conditions that apply to the opened file. For example, a condition could be that the file can be read but not updated (read-only). | Creating a Btrieve File Tasks |

| Client ID  | If you want this login to apply to a specific client ID, click Use and specify the range. Otherwise, leave as is. | Creating a Btrieve File Tasks |

| Sharing    | The database engine ignores the “Sharing” options. The “Sharing” options apply only to a legacy version of the database engine, Btrieve 6.15. | |

Create a File Dialog GUI Reference (Advanced)

This dialog box allows you to specify additional characteristics for the file being created. Click an area of the image for which you want more information.
Function Executor Graphical User Interface

Table 66  Function Executor Create File Dialog (Advanced)

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key and Segment commands</td>
<td>Allow you to add or delete a key or to add, insert, or delete a key segment.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>File Specifications</td>
<td>Displays and allows you to modify characteristics of the file.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>Statistics</td>
<td>Provides read-only information about the file.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>Key and segment matrix</td>
<td>Displays the keys and segments for the file, listing the starting position, length, attributes, and data type for them. Clicking a row allows you to see, and modify if you want, information in the Key and Segment blocks.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>Key</td>
<td>Displays and allows you to modify characteristics of a key.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>Segment</td>
<td>Displays and allows you to modify characteristics of a key segment.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
<tr>
<td>Command buttons</td>
<td>Allow you create a data file, a description, or to cancel the dialog box.</td>
<td>Creating a Btrieve File Tasks</td>
</tr>
</tbody>
</table>

Transactions

This dialog box allows you to control transactions during your Function Executor session. Click an area of the image for which you want more information.

Table 67  Function Executor Transactions Dialog

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main application window</td>
<td>Main features of the program.</td>
<td>Application Window</td>
</tr>
<tr>
<td>Transaction type</td>
<td>Specifies whether you want an exclusive or concurrent transaction.</td>
<td></td>
</tr>
<tr>
<td>Lock information</td>
<td>Specifies the type of locking you want in the transaction.</td>
<td></td>
</tr>
<tr>
<td>Start transaction button</td>
<td>Starts a transaction.</td>
<td></td>
</tr>
<tr>
<td>Commit transaction button</td>
<td>Commits the active transaction.</td>
<td></td>
</tr>
<tr>
<td>Abort transaction button</td>
<td>Cancels the active transaction and rolls back any changes made.</td>
<td></td>
</tr>
</tbody>
</table>
Testing Btrieve Operations

Table 67  Function Executor Transactions Dialog  continued

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>File area</td>
<td>This area contains one or more open files.</td>
<td>Main Window</td>
</tr>
<tr>
<td>Transaction indicator</td>
<td>Indicates whether you are currently in a transaction. If you are in a transaction, this display will show either Concurrent or Exclusive.</td>
<td></td>
</tr>
</tbody>
</table>

**File Statistics**

This dialog box allows you to see the statistics for a Btrieve file. Click an area of the image for which you want more information.

![File Statistics](image)

Table 68  Function Executor File Statistics Dialog

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File information</td>
<td>Displays statistical information about the file.</td>
</tr>
<tr>
<td>Flags</td>
<td>Allows you to view the flags set for this file. Any flag that is set will have a check mark next to its listing in the dialog box.</td>
</tr>
<tr>
<td>Keys</td>
<td>Allows you to view the keys defined in this file.</td>
</tr>
<tr>
<td>Key legend</td>
<td>Describes the single letter values that indicate Key attributes for the file.</td>
</tr>
<tr>
<td>Print button</td>
<td>Allows you to print the file statistics. Set up the printer using the File menu.</td>
</tr>
</tbody>
</table>
History

This dialog box allows you to see all the operations you have performed. Click an area of the image for which you want more information.

Table 69  Function Executor History Dialog

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>File menu</td>
<td>Allows you to perform the following operations:</td>
<td>• To toggle the docking status of the History window</td>
</tr>
<tr>
<td></td>
<td>• Import a history file</td>
<td>• To toggle the Always On Top status of the History window</td>
</tr>
<tr>
<td></td>
<td>• Export a history file</td>
<td>• To reset the History window to default settings</td>
</tr>
<tr>
<td></td>
<td>• Close the history window</td>
<td></td>
</tr>
<tr>
<td>Settings menu</td>
<td>Allows you to specify the visual attributes of the History window.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Save on Exit - specifies whether you want your customization of the History window to be saved between sessions.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Defaults - Resets the History window to default settings. This removes any settings you specified.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Docked - Toggles the state of the History window between a separate window and one that is attached to the Application Window.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Stay On Top- toggles the state of the History window between one that can lose focus and one that cannot.</td>
<td></td>
</tr>
<tr>
<td>Execute command</td>
<td>Loads the History Playback window</td>
<td></td>
</tr>
</tbody>
</table>
Testing Btrieve Operations

Table 69  Function Executor History Dialog

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of operations</td>
<td>Lists operations that you have recently performed. Each operation is logged with the following information:</td>
<td>History Tasks</td>
</tr>
<tr>
<td></td>
<td>• FileID</td>
<td>History</td>
</tr>
<tr>
<td></td>
<td>• Operation name or number depending on the status of the OpCodeNames check box.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Status code when that operation was performed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Number of times that operation was performed</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Key number set for the operation.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Contents of the key buffer.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Contents of the data buffer.</td>
<td></td>
</tr>
<tr>
<td>Logging</td>
<td>Toggles the inclusion of future operations in the history list.</td>
<td></td>
</tr>
<tr>
<td>OpCode Names</td>
<td>Toggles the display in the Operation column between operation code names (such as B_OPEN) and operation code numbers (such as 0 for B_OPEN)</td>
<td></td>
</tr>
</tbody>
</table>
Function Executor Tasks

Function Executor tasks are grouped into the following categories:

- Starting Function Executor Tasks
- Performing Operations Tasks
- Opening a File Tasks
- Creating a Btrieve File Tasks
- History Tasks

Starting Function Executor Tasks

➢ To start the Function Executor utility

1. Access Function Executor from the operating system Start menu or Apps screen or from the Tools menu in PSQL Control Center.
2. The main window (Figure 13) appears.

Performing Operations Tasks

Because Btrieve provides many operations, this chapter cannot explain them all. The following sections discuss some common operations as well as some new ways of performing them with the Function Executor.
Note Selecting options from all menus performs the intended operation immediately. It does not fill in the grid and wait for you to execute the command as in previous versions. Also, closing the form closes each open file. No longer do you need to manually perform the close operation.

General Operations-Related Tasks

- To get help for a status code

For other tasks, see these sections:

- Opening a File Tasks
- Creating a Btrieve File Tasks
- History Tasks

To get help for a status code

1. When a status code is received using Function Executor, it is displayed on the status bar of the open file.

   Move your mouse so it hovers over the status code that is displayed in red.

   Figure 14 Status Code Received

2. If you click the status code indicator, the full documentation for the status code is displayed as shown in Figure 15.

   Figure 15 Status Code Documentation

41: The MicroKernel does not allow the attempted operation

The Microkernel returns this status code for the following reasons:

- The application tried to perform an operation that is not allowed under these operating conditions. The Microkernel does not allow some operations under certain operating conditions. For example, the Microkernel returns this status code if the application attempts to perform a Step operation on a key-only file.
- If using a server engine, the key number parameter of a continuous operation MicroKernel call is not valid.

The Microkernel prohibits certain operations during transactions because they have too great an effect on the file or on performance. These operations include Set Owner, Clear Owner, Create Index, and Drop Index.
Opening a File Tasks

➢ To open a data file with Function Executor

1. From the File menu, select Open. The following dialog box appears:

![Open Btrieve File Dialog Box](image)

**Note** Client ID: If you have Use disabled, Function Executor will use a BTRV() function call. If Use is enabled, it will use a BTRVID() function call for each operation you execute on this file. With Auto enabled, Function Executor will generate a client ID for you. If you have Auto disabled, then you may enter values manually.

2. Click Browse.

3. Double-click the desired file name.

Other Ways to Open a File with Function Executor

1) You can drag a file from Windows Explorer or from an operating system folder view into the Open dialog box. This step fills in the filename for you.

2) You can drag one or more files into the main window.

3) After opening one file (so you have the editor window available), you can use the OpCode 0 to open another file. The file will appear in a new window.

4) You can run Function Executor from the DOS command line and specify a list of filenames to open. For example, you could use the following command line to open two files from the DEMODATA sample database:

```
WBExec32 person.mkd billing.mkd
```

You can even use wildcard characters, as in the example:

```
WBExec32 *.mkd
```

Running this command allows you to associate file extensions (types) with Function Executor. For example, you can associate MKD, BTR, DAT, or any other extension with Function Executor. Thus, when you double-click the file in Explorer, it automatically opens the file with Function Executor.
Creating a Btrieve File Tasks

There are two options in creating a Btrieve file with Function Executor. If a file is already open, you can clone it; otherwise you can start from scratch.

Caution

In the same directory, no two files should share the same file name and differ only in their file name extension. For example, do not name a data file Invoice.btr and another one Invoice.mkd in the same directory. This restriction applies because the database engine uses the file name for various areas of functionality while ignoring the file name extension. Since only the file name is used to differentiate files, files that differ only in their file name extension look identical to the database engine.

Method 1: Using a current file as a template

1. From the File menu, select New. The following dialog box will appear:

   ![Modify File Definition Dialog Box]

2. You can manipulate keys from this dialog box as well. You can Add, Create, or Insert Segments from the Key menu. You can also save the new file as a description for use with BUtil create. Select Save As Desc and indicate the name and location where you would like the file saved.

3. To create the file, click Create. This will open the file and display a message indicating success.

Method 2: Creating a new file from scratch

1. Click the Create icon on the main toolbar; or, if no file is open yet, you may click File and then New, as before.

2. If a file is already open on screen, a drop down box will appear. Choose Create New File from Scratch.

3. The same dialog box as before will appear, but it will be blank - allowing you to input brand new values.
4. Start by adding a new Key using the Key menu, or press Ctrl-A.
5. Fill in the attributes for the key in the lower section of the dialog box.
6. Continue adding or removing new keys and segments as desired, using the menus or right-clicking the key in the list.
7. Now click the Create button to execute the B_Create (14) operation. This will automatically open the file on screen as well.

**History Tasks**

The following tasks are related to the History feature:

- To display the History window
- To toggle the docking status of the History window
- To toggle the Always On Top status of the History window
- To reset the History window to default settings

> To display the History window

1. Click View > History or click the History button.

> To toggle the docking status of the History window

1. If the History window is not visible, display it. (see To display the History window).
2. In the history items window, right-click the display and check or clear the Docked option as shown in the following figure:

![Figure 18 - Undocking the History window](image)

When docked, the History window is connected to the application window as shown in Figure 18. When not docked, the History window is a distinct window. When undocked, the History window has menu items that duplicate the commands seen from the right-click menu in Figure 18.

> To toggle the Always On Top status of the History window

1. If the History window is not visible, display it (see To display the History window).
2. This feature only applies to the History window when it is in the undocked state (see To toggle the docking status of the History window).
3 In the history items window, right-click the display and check or clear the Stays On Top selection.

➢ To reset the History window to default settings
1 If the History window is not visible, display it (see To display the History window).
2 In the history items window, right-click the display and select Settings > Defaults.
Manipulating Btrieve Data Files with Maintenance

Handling Btrieve Files with the Maintenance Utility

This chapter discusses the following topics:

- Maintenance Utilities Overview
- Btrieve Interactive Maintenance Utility
- File Information Editor
- Owner Names
- Statistics Report
- Indexes
- Data
- Btrieve Command-Line Maintenance Utility (butil)
- Importing and Exporting Data
- Creating and Modifying Data Files
- Viewing Data File Statistics
- Displaying MicroKernel Engine Version
- Unloading the MicroKernel Engine and Requester (DOS only)
- Performing Continuous Operations
- Performing Archival Logging
Maintenance Utilities Overview

PSQL provides both an interactive Maintenance utility and a command-line Maintenance utility. Both Maintenance utilities perform the following common file and data manipulations:

- Create new data files based on file and key specifications you define.
- Provide file and key specifications for existing data files.
- Set and clear owner names for data files.
- Create and drop indexes on data files.
- Import and export ASCII sequential data.
- Copy data between PSQL data files.
- Recover changes made to a file between the time of the last backup and a system failure.

While both utilities provide the same core functionality, minor differences exist. For example, the interactive Maintenance utility allows you to create description files based on file and key specifications you define. The command-line Maintenance utility allows you to start and stop continuous operation on a file or set of files locally on the server.

Before you use either Maintenance utility, you should be familiar with Btrieve fundamentals, such as files, records, keys, and segments. For information about these topics, refer to the PSQL Programmer’s Guide.

Note The PSQL product provides two categories of maintenance utilities: Btrieve and SQL. The SQL Maintenance Utility supports data source names (DSNs), which are used for relational access through ODBC.
Btrieve Interactive Maintenance Utility

The Interactive Maintenance utility is a Windows application that runs on Windows 32-bit and 64-bit platforms. Use this utility if you prefer a graphical interface or if you want to create a description file. This section contains the following major topics:

- File Information Editor
- Owner Names
- Statistics Report
- Indexes
- Data

Each major topic contains tasks specific to that topic.

Extended File Support

The size of a MicroKernel data file can be larger than the operating system file size limit. When you export data from an extended MicroKernel file to an unformatted file, the size of the unformatted file can exceed the database engine file size limit because of the differences in the physical format.

When you are exporting large files, the Interactive Maintenance utility detects when the unformatted file exceeds a 2 GB file size limit and starts creating extension files. This process is transparent. Extension files and the original unformatted file must reside on the same volume. (The size limit for a file varies depending on the operating system and file system. The 2 GB size is simply the limit enforced by the database engine.)

The extension file uses a naming scheme in which the file names are similar to the base file name. In contrast to native MicroKernel Engine extension files which use a caret “^” to indicate extension file status, the unformatted extension files use a tilde “~” to avoid overwriting any existing extended engine files with the same base file name. The first export extension file is the same base file name with “.~01” extension. The second extension file is “.~02,” and so on. These extensions are appended in hexadecimal format.

The naming convention supports up to 255 extension files, thus supporting files as large as 256 GB. Additionally, when you import data from an unformatted file, the utility detects whether the file has extensions and loads the data from the extension file.

Long File Names and Embedded Spaces Support

Long file name support, including support for embedded spaces is available in all supported operating system environments. All references to files can contain embedded spaces and be longer than 8 bytes.

Older versions of Btrieve allowed spaces to be added at the end of a file name in path-based operations such as Open and Create. This is still the default behavior. Existing applications will not break. However, if you want to take advantage of file and directory names with embedded spaces, set the Embedded Spaces configuration setting for the client requester to On. Note that On is the default setting.

Even when you turn the option off an application that accesses a file having a name with embedded spaces can enclose that name in double quotes while making the BTRV/BTRVID/BTRCALL/BTRCALLID call to open or create the file.
**Record and Page Compression**

PSQL provides two types of data compression: record and page. These two types may be used separately or together. The primary purpose for both compression types is to reduce the size of the data files and to provide faster performance depending on the type of data and on the type of data manipulation.

**Record Compression**

Record compression requires a file format of 6.0 or later. Record compression can result in a significant reduction of the space needed to store records that contain many repeating characters. The database engine compresses five or more of the same contiguous characters into 3 bytes.

When creating a file, the database engine automatically uses a page size larger than what is specified to allow room for the specified record length. If the uncompressed record length is too large to fit on the largest available page, the database engine automatically turns on record compression.

Because the final length of a compressed record cannot be determined until the record is written to the file, the database engine creates a file with record compression as a variable-length record file. Compressed images of the records are stored as variable-length records. Individual records may become fragmented across several file pages if your application performs frequent insertions, updates, and deletions. The fragmentation can result in slower access times because the database engine may need to read multiple file pages to retrieve a single record.


**Page Compression**

Page compression requires a file format of 9.5 or later. Internally, a PSQL data file is a series of different types of pages. Page compression controls the compression and decompression of data pages within a file.

As a file is read from physical storage, data pages are decompressed and held in a memory cache. Record reads and updates are performed against the uncompressed data in the memory cache. When a write action occurs, the data page is compressed then written to physical storage. Depending on cache management, the compressed page is also retained in memory until accessed again.

If the type of data cannot be significantly compressed, the database engine writes the data to physical storage uncompressed.
Deciding When To Use Compression

The benefits obtained by using record compression, page compression, or both depends entirely on the type of data being compressed. Given that, the following table discusses some general factors to consider when deciding to use data compression or not.

<table>
<thead>
<tr>
<th>Record</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

**Factors to Consider**

- Record compression is most effective for the following conditions:
  - Each record has the potential for containing a large number of repeating characters. For example, a record may contain several fields, all of which may be initialized to blanks by your task when it inserts the record into the file. Record compression is more efficient if these fields are grouped together in the record, rather than being separated by fields containing other values.
  - The computer running the database engine can supply the extra memory required for compression buffers.
  - The records are read much more frequently than they are changed.

  If the fixed length portion of a record is longer than the page size minus overhead, compression is used automatically.

  Note that you cannot use record compression for key-only files or for files that use blank truncation.

- Page compression is most effective for the following conditions:
  - Data is highly compressible using a ZIP-type compression algorithm. When the file size can be significantly decreased because of page compression, such as 4 to 1 compression, file performance can be increased significantly.
  - The pages are read much more frequently than they are inserted, updated, or deleted.

  Note that the database engine writes data pages to physical storage uncompressed if the data cannot be significantly compressed.

- The use of record compression and page compression is most effective when records contain a large proportion of blank space and the pages are read much more frequently than they are inserted, updated, or deleted.

The Btrieve Maintenance Utility Interface

Access **Maintenance** from the operating system **Start** menu or **Apps** screen or from the **Tools** menu in PSQL Control Center. The Maintenance main window looks like the following.
Menu Options

The interactive Maintenance utility provides the following menus:

- **Options**: Allows you to display the File Information Editor, set and clear owner names, generate statistics reports, and exit the utility.
- **Index**: Allows you to create and drop indexes.
- **Data**: Allows you to load data from ASCII files, save data to ASCII files, copy records between data files, and perform a roll forward operation to recover changes made to a data file between the time of the last backup and a system failure.
- **Help**: Provides access to the Maintenance utility help system.

Getting Help

To access the Maintenance utility help system, click **Help** in the dialog box for which you want help, or choose a command from the **Help** menu, as follows:

- **Contents**: Provides a list of Maintenance utility help topics.
- **About**: Displays copyright information and the product version number.
File Information Editor

This section provides general information about the File Information Editor with which you can create new files based on file and key specifications you construct. Because this Editor allows you to load information based on an existing file, it is also useful for viewing file and key specifications on existing data files. You can also create a new file based on the file and key specifications of an existing file (similar to the CLONE command for BUTIL, the command-line Maintenance utility).

**Caution** No two files can share the same file name and differ only in their file name extension if both files are in the same directory. For example, do not name a data file Invoice.btr and another one Invoice.mkd in the same directory. This restriction applies because the database engine uses the file name for various areas of functionality while ignoring the file name extension. Since only the file name is used to differentiate files, files that differ only in their file name extension look identical to the database engine.

Open the File Information Editor by clicking **Options > Show Information Editor**.

**File Information Editor Dialog Elements**

At the top of the Editor, the following buttons appear:

- **Load Information**: Loads information based on an existing file. When you load information, you are not editing the existing file. Instead, you are loading a copy of the information about that file.
- **Create File**: Creates a new file based on current information in the dialog box.
- **Set To Defaults**: Sets the controls to default values.
- **Description Comments**: If you are creating a description file, allows you to enter notes about the file.
- **Help**: Displays help for the File Information Editor dialog box.
Data File Info

The **Data File Info** area, also at the top of the File Information Editor, contains the following controls:

- **Owner Name**: Provides a text box you can use to specify the owner name, if applicable, for an existing file.
- **Version**: Earliest version of the database engine that can read all the attributes of the file. For example, if you created a file using the 9.5 database engine but did not use any attributes specific to 0.5, the Maintenance utility displays 9.0 as the version number. See File Version Notes for additional information about file format versions.
- **Total Records**: Total number of records in the file.

File Specification

The **File Specification** area is in the middle of the File Information Editor. Table 71 describes the controls in this box.

### Table 71  File Specification Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Range</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Record Length</strong></td>
<td>Specifies the logical data record length (in bytes) of the fixed-length records in a file. For information about record length and overhead, see “Record Length” in <em>PSQL Programmer’s Guide</em>, which is part of the PSQL Developer Reference.</td>
<td>Minimum is 4 bytes. Maximum is variable. If the record length specified exceeds the page size minus overhead, the database engine automatically tries the next available page size for the file format. If the record length exceeds the maximum page size minus overhead, the engine turns on record compression.</td>
<td>100</td>
</tr>
<tr>
<td><strong>Page Size</strong></td>
<td>Specifies the physical page size (in bytes) for the file.</td>
<td>512 – 4096 for file versions prior to 9.0 (a multiple of 512 bytes up to 4096) 512, 1024, 1536, 2048, 2560, 3072, 3584, 4096, or 8192 for file version 9.0. 1024, 2048, 4096, 8192, or 16384 for file versions 9.5 and newer.</td>
<td>4,096</td>
</tr>
<tr>
<td><strong># Keys</strong></td>
<td>Indicates the number of distinct keys (as opposed to key segments) currently defined in the Editor. Reflects the number of keys in the Key list.</td>
<td>0 – 119</td>
<td>0</td>
</tr>
<tr>
<td><strong># Segments</strong></td>
<td>Indicates the number of key segments currently defined in the Editor. Reflects the number of segments in the Segment list.</td>
<td>0 – 119 for file versions prior to 9.5. 0 – 420 for file versions 9.5 and newer. Note that, for all file versions, the maximum number of segments for the Relational Engine is 119.</td>
<td>0</td>
</tr>
<tr>
<td><strong>Available Linked Keys</strong></td>
<td>Specifies how many 8-byte place holders you want to reserve for future linked-duplicatable keys. If you are loading information based on an existing data file, this value reflects the number of place holders currently available in that file. (The number of originally reserved place holders is not stored in the file.)</td>
<td>0 – 119</td>
<td>3</td>
</tr>
</tbody>
</table>
Table 71  File Specification Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Range</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key-Only</td>
<td>Indicates whether the file is key-only. Not applicable if you turn Record Compression on, if you turn Variable Records on, or if you define more than one key for the file.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td>Balanced Indexing</td>
<td>Specifies that the file uses the balanced indexing method of managing key pages.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td>Pre-allocation</td>
<td>Specifies that the file uses preallocated pages.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td># Pages</td>
<td>Specifies the number of pages you want preallocated when you create the file. Applicable only if Pre-allocation is turned on. If you are loading information based on an existing data file, this value reflects the number of unused, preallocated pages left in that file. (The number of originally preallocated pages is not stored in the file.)</td>
<td>1 – 65,535</td>
<td>0</td>
</tr>
<tr>
<td>Record Compression</td>
<td>Specifies that the file uses record compression. Not applicable for key-only files or files that use blank truncation. See also Record and Page Compression.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td>Page Compression</td>
<td>Specifies that the file uses page compression. See also Record and Page Compression.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td>Variable Records</td>
<td>Specifies that the file can contain variable-length records.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td>Blank Truncation</td>
<td>Specifies whether the file uses blank truncation on variable records to conserve disk space. Applicable only if Variable Records is turned on.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td>Include VATs</td>
<td>Specifies whether the file supports Variable-tail Allocation Tables for faster access to data in very long records. Applicable only if Variable Records is turned on.</td>
<td>On or Off</td>
<td>Off</td>
</tr>
<tr>
<td>% Free Space</td>
<td>Specifies the amount of unused space a file’s variable pages must have available before the database engine creates a new variable page. Applicable only if Record Compression or Variable Records are turned on.</td>
<td>5, 10, 20, or 30</td>
<td>5</td>
</tr>
</tbody>
</table>

**Key**

At the bottom left in the dialog box is the **Key** group box. Table 72 describes the controls in this area. These controls are specific to the key highlighted in the **Key** list, not just to the current key segment.
When you change the setting for one of these controls, the change affects all segments of the specified key.

Table 72  Key Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duplicates</td>
<td>Specifies that the key can have duplicate values (linked duplicates).</td>
<td>On</td>
</tr>
<tr>
<td></td>
<td>See Methods for Handling Duplicate Keys.</td>
<td></td>
</tr>
<tr>
<td>Modifiable</td>
<td>Specifies that the key value can be modified after creation. Allowing modification of key values does not affect performance. Key pages are only updated if the actual key value changes, not if non-key fields in a particular record are changed.</td>
<td>On</td>
</tr>
<tr>
<td>Repeating Duplicates</td>
<td>Specifies that the database engine uses the repeating duplicates method of storing duplicate key values.</td>
<td>Off</td>
</tr>
<tr>
<td></td>
<td>See Methods for Handling Duplicate Keys.</td>
<td></td>
</tr>
<tr>
<td>Sparse Key (Null Key)</td>
<td>A sparse key contains fewer key values than the number of record in the file. To specify which key values are excluded from the index, see the next two controls. Applicable only to keys that contain nullable segments.</td>
<td>Off</td>
</tr>
<tr>
<td>All Segments (Null)</td>
<td>Specifies that if all key segments in the record contain a null value, the database engine does not include that record in the index. Applicable only if Sparse Key (Null Key) is turned on. Equivalent to key flag 0x0008. Whether a segment is evaluated as null is determined solely by the null indicator segment for that field; the contents of the field are not evaluated.</td>
<td>Off</td>
</tr>
<tr>
<td>Any Segment (Manual)</td>
<td>Specifies that if one or more key segments contains a null value, the database engine does not include that record in the index. Applicable only if Sparse Key (Null Key) is turned on. Equivalent to key flag 0x0200. Whether a segment is evaluated as null is determined solely by the null indicator segment for that field; the contents of the field are not evaluated.</td>
<td>Off</td>
</tr>
<tr>
<td>ACS Information</td>
<td>Allows you to specify an alternate collating sequence (ACS) for the key. Applicable only if the Use ACS check box is selected for a segment of the key.</td>
<td>Off</td>
</tr>
<tr>
<td>Unique Values</td>
<td>Indicates the number of unique key values in the file. Applicable only if you are loading information based on an existing data file.</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Key List and Segment List

At the bottom middle of the dialog box, the Key list shows the key numbers defined in a file. (For 6.x and later files, these key numbers do not have to be consecutive; they can have gaps between them.) The Maintenance utility displays the highlighted key's specifications in the Key box at the bottom left of the dialog box.

Also at the bottom middle of the dialog box, the Segment list shows the key segment numbers defined for the key highlighted in the Key list. The Maintenance utility displays the highlighted segment's specifications in the Segment box at the bottom right of the dialog box.

In addition, the following buttons appear under the Key and Segment lists:

- **Insert**: Defines a new key or segment.
- **Delete**: Removes the highlighted key or segment specification.
- **Compress**: Renumbers the keys consecutively. You can use this button to remove gaps that result from deleting a key specification.
Because these buttons control key specifications for a file you want to create, you cannot use them to operate on keys in an existing file. If you want to create or drop an index on an existing file, refer to **Index Tasks**.

**Key Segment**

At the bottom right in the dialog box is the **Key Segment** group box. Table 73 describes the controls in this area. These controls are specific to the segment highlighted in the **Segment** list.

**Table 73  Key Segment Specification Controls**

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Type</td>
<td>Specifies a data type for the key segment. The NULL data type indicates that the index is one byte Null indicator segment. It must be in a multisegment key and it must precede another key segment that is not a NULL type. The number used in the Btrieve API for this key type is 255.</td>
<td>String</td>
</tr>
<tr>
<td>Position</td>
<td>Specifies by number the relative starting position of the beginning of this key segment in the record. The value cannot exceed the record length.</td>
<td>1</td>
</tr>
<tr>
<td>Length</td>
<td>Specifies the length (in bytes) of the key segment. This value cannot exceed the limit dictated by the data type for the segment. The total of key position and key length cannot exceed the record length.</td>
<td>10</td>
</tr>
<tr>
<td>Null Value (Hex)</td>
<td>Specifies the null character value (in hexadecimal) for the key segment. Applicable only if the <strong>Null Key</strong> check box is selected for the key.</td>
<td>Binary zero</td>
</tr>
<tr>
<td>Case Insensitive</td>
<td>Specifies whether the segment is sensitive to case. Applicable only for STRING, LSTRING, and ZSTRING data types or for keys that do not use an ACS.</td>
<td>On</td>
</tr>
<tr>
<td>Descending</td>
<td>Specifies that the database engine sort the key segment values in descending order (that is, from highest to lowest).</td>
<td>Off</td>
</tr>
<tr>
<td>Use ACS</td>
<td>Specifies that the segment uses the alternate collating sequence defined for the key. Applicable only for <strong>string</strong>, <strong>lstring</strong>, and <strong>zstring</strong> data types that are case sensitive.</td>
<td>Off</td>
</tr>
<tr>
<td>NULL Value Discrete Ordering</td>
<td>NULL Value Discrete Ordering is used for the null indicator segment (NIS) to determine whether the MicroKernel Engine should treat the NIS as a boolean value, where any non-zero value is considered NULL, or as a one byte integer, where zero is considered non-null and all other values are considered different types of null. In this case they are sorted as discrete values. The Btrieve API uses the NO_CASE flag, 0x0400, to indicate discrete ordering should be performed, because that flag was previously unused for integer values.</td>
<td>Off</td>
</tr>
</tbody>
</table>

**Methods for Handling Duplicate Keys**

Multiple records may carry the same duplicated value for index keys. The two methods to keep track of the records with duplicate key values are called linked duplicates and repeating duplicates.

**Linked Duplicates**

The linked duplicates method uses a chain technique in which each record in the group connects to its neighbors by means of pointers. Each entry on an index page contains a pair of record pointers that indicate the first and last links in the chain of records that duplicate that key's value. This makes each key page entry 4 bytes longer than a repeating duplicates index. In addition, each record on the data page requires an extra 8 bytes of overhead for each linked duplicates index. These 8 bytes consist of two record pointers that point to the next and previous records in the chain.
The first record pointer holds the address of the first, or oldest, record stored. The second pointer holds the address of the most recent, or newest record. After the first record is written but before any others are added, both pointers on the key page entry hold the first record's address. Subsequent records cause the second pointer to be changed to point to each record as it is added. This permits the record pointer for the last record to be used as the previous-record link of the chain built in the data page when the record is added, and also to be used to locate that previous record.

Repeating Duplicates

With the repeating duplicates method, each duplicate key value is stored on both the index page and within the record on the data page. Each key value has only one record pointer instead of two. This method requires no chaining within the data records and saves the 8 bytes of overhead per index within each record. Since the key value is repeated for each duplicate record, the indexes affected increase in size.

Method Comparisons

The linked duplicates and repeating duplicates methods can be compared based on the following criteria:

- Ordering
- Storage
- Performance
- Concurrency

Ordering

A linked duplicates index retrieves duplicates in the order in which they were inserted. A repeating duplicates index retrieves duplicates in the order in which they are located within the file. Since location with a file cannot be controlled, the ordering must be considered as random.

Storage

A linked duplicates index requires 12 more bytes for the first occurrence of each duplicate key value. That includes 8 extra bytes on each record and 4 extra bytes for the key page entry. But each duplicate record requires no additional space in the key page, and adds only 8 bytes per record. Therefore, as the number of duplicates per key value increases, and as the size of the key value increases, linked duplicate indexes can save significant storage space used by key pages. However, storage space can increase if your file contains very few records with duplicate keys, the key length is very short, or both.

The following figure exemplifies the amount of storage space saved using linked duplicate indexes. Note that linked duplicate indexes take more space if duplicate records per key value are few. As the number of duplicate records per key value increases, however, linked duplicate indexes require less pages, providing significant space savings.
Performance

Faster performance results when fewer pages are involved in an index search because fewer pages must be read from disk. The linked duplicates method generally uses less physical storage space and therefore provides faster performance. The repeating duplicates method provides a performance advantage if only a small number of keys have duplicates.

Concurrency

The database engine provides page-level concurrency when several concurrent transactions are active on the same file at the same time. This applies to most changes to key pages and for all changes to data pages. The concurrency means that the same page can contain pending changes from separate transactions at the same time, and the transactions can be committed in any order. Repeating duplicate indexes take the most advantage of this concurrency.

Linked duplicate indexes add another limitation on concurrency that does not exist with repeating duplicates. When a new duplicate is created, the new record is linked to another record at the end of the list. This record linking causes two records to be locked instead of one. Since all duplicates are added to the end of the chain of linked records, only one duplicate can be inserted at a time.

Such a record lock conflict usually causes other clients to wait until the first transaction is committed. In a concurrent environment, if all new records use the same duplicate value, then concurrency can effectively be reduced to one transaction at a time. And if transactions are large or long lasting, this serialization can affect performance tremendously.

Performance is typically better if you use repeating duplicate indexes for databases that are updated in a concurrent environment. Therefore, unless you have a compelling reason to use the linked duplicates
Manipulating Btrieve Data Files with Maintenance

method, you should use repeating duplicate indexes for databases that are updated in a concurrent environment.

**Information Editor Tasks**

You perform the following tasks with the File Information Editor:

- Loading Information from an Existing Data File
- Creating a New File
- Compacting Btrieve Data Files
- Specifying a Key’s Alternate Collating Sequence

**Loading Information from an Existing Data File**

When you load information from an existing file, you are not editing the existing file. Instead, you are loading a copy of the information about that file. Generally, you want to load a data file before performing other tasks with the File Information Editor, but this is not mandatory.

1. **To load information from an existing data file into the File Information Editor**
   
   1. Click **Load Information** at the top of the File Information Editor. The **Select File** dialog box appears.

   ![Select File Dialog Box](image)

   2. Specify the name and path of the file for which you want to load information. (By default, data files have the `.mkd` extension.)

   The Maintenance utility first attempts to open the specified file as a data file. If the file requires an owner name, the utility prompts you for one. (Because owner names are optional, the file you open may not require an owner name.) If the specified file is not a data file, the utility then attempts to open the file as a description file.

**Creating a New File**

You can create a new file based on the current information in the File Information Editor or on new information you provide.

1. **To create a new file based on the current information in the File Information Editor**
   
   1. Click **Create File** at the top of the File Information Editor dialog box. The **Create File** dialog box appears.
2 Specify the controls in the **Create File** dialog box, which are described in Table 74.

### Table 74  Create File Dialog Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Name</td>
<td>Specifies a name and path for the file. By default, data files have the .mkd extension.</td>
<td>N/A</td>
</tr>
<tr>
<td>File Type</td>
<td>Specifies the type of file to create. If you are creating a description file, you can use the Index Only option, which creates a description file you can use with the BUTIL utility to add an index to an existing data file. (For more information, refer to Creating Indexes.)</td>
<td>MicroKernel-compatible</td>
</tr>
<tr>
<td>System Data</td>
<td>Determines whether the utility includes system data in the file. If you choose Use Engine Setting, the utility uses the setting for the System Data configuration option described. If you choose No System Data, the utility does not create system data, regardless of the engine configuration. If you choose Force System Data, the utility creates system data, regardless of the engine configuration. This is applicable only if the file type is MicroKernel-compatible.</td>
<td>Use Engine Setting</td>
</tr>
</tbody>
</table>

**Adding Comments to a Description File**

The comments are written to the top of the description file when you create the description file. For example, the comment, “This is my file,” appears at the top of the description files as /* This is my file */. If you add additional comments after creating the description file, you need to create the file again to include the additional comments.

- **To add comments to a description file**

1 Click **Description Comments**. The **Description File Comments** dialog box appears.
2 Enter a block of comments up to 5,120 characters long.
3 Click OK when you are finished entering comments.

**Compacting Btrieve Data Files**

You can compact a Btrieve data file to remove unused space in it, which typically decreases the file size. You can also perform this procedure using the command-line Maintenance utility (see To compact a Btrieve data file).

➢ To compact a Btrieve file
1 Click Load Information in the File Information Editor and select the file you want to compact.
2 Click Create File, give the file a new name (which creates a clone) in the Create File dialog box, and click OK.
3 From the Data menu on the main window, select Save. In the Save Data dialog box, enter the name of the original file in the From MicroKernel File box and then specify a name for the output file (for example, <original file>.out) in the To Sequential File box.
4 Click Execute. The Save Data dialog box displays the results of the save. Click Close.
5 From the Data menu, select Load. In the Load Data dialog box, enter the name of the sequential data file you just saved in the From Sequential File box. Then enter the name of the clone file you created in Step 2 in the To MicroKernel File box.
6 Click Execute. The Loading Data dialog box displays the results of the load. Click Close.

You can now compare the size of the original file to the clone file to verify the reduction in size.

**Specifying a Key’s Alternate Collating Sequence**

You can use an alternate collating sequence (ACS) to sort string keys (types STRING, LSTRING, and ZSTRING) differently from the standard ASCII collating sequence. By using one or more ACSs, you can sort keys as follows:

- By your own user-defined sorting order, which may require a sorting sequence that mixes alphanumeric characters (A-Z, a-z, and 0-9) with non-alphanumeric characters (such as #).
- By an international sorting rule (ISR) that accommodates language-specific collations, including multibyte collating elements, diacritics, and character expansions and contractions.
Files can have a different ACS for each key in the file, but only one ACS per key. Therefore, if the key is segmented, each segment must use either the key’s specified ACS or no ACS at all. For a file in which a key has an ACS designated for some segments but not for others, Btrieve sorts only the segments that specify the ACS.

The ISR tables are provided with PSQL and are based on ISO-standard locale tables. ISR tables are stored in the COLLATE.CFG file, which is installed with the PSQL database engine. Multiple data files can share a single ISR.

➢ To specify a key’s alternate collating sequence

1. Click ACS Information.

   The Maintenance utility displays the Specify ACS Information dialog box.

   Figure 25 Specify ACS Information Dialog Box

2. You can specify either a Country ID and Code Page, an ACS File name, or an International Sorting Rule (ISR) as follows:

   Table 75 ACS Information Controls

<table>
<thead>
<tr>
<th>Control</th>
<th>Description</th>
<th>Default</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS Country/Code</td>
<td>An Intel-format number that identifies your country. Refer to your operating system’s documentation for specific information.</td>
<td>-1</td>
</tr>
<tr>
<td>Country ID</td>
<td>An Intel-format number that identifies the code page you want to use. Refer to your operating system’s documentation for specific information.</td>
<td>-1</td>
</tr>
<tr>
<td>Code Page</td>
<td>Specifies the fully qualified file name of the alternate collating sequence file.</td>
<td>N/A</td>
</tr>
<tr>
<td>International Sorting Rule</td>
<td>When you click this radio button you can specify a specific ISR table for sorting international data. PSQL provides a set of pregenerated ISR tables, which are listed in the PSQL Programmer’s Guide.</td>
<td></td>
</tr>
</tbody>
</table>

3. When you specify a Country ID and Code Page ID, the database engine stores the locale-specific collating sequence in the data file. Moreover, the database engine can insert new key values correctly, even if the locale changes.
4 When you specify an ACS filename for a data file, the database engine copies the contents of the ACS file into the data file. (That is, the data file does not contain the file name of the ACS file.) The ACS identifies itself using an eight-digit name (such as UPPER). Subsequently, when you view the ACS information for a data file, the Maintenance utility displays this eight-digit name, not the file name of the original ACS.

5 When you specify an ACS file for a description file, the Maintenance utility copies the actual path and file name of the ACS file into the description file. Subsequently, when you view the ACS information for a description file, the Maintenance utility attempts to locate the specified ACS file.

To specify an ACS that sorts string values using an ISO-defined, language-specific collating sequence, you must specify an ISR table name. The Table Name field is limited to 16 characters. For more information on ISRs, refer to the PSQL Programmer’s Guide in the Developer Reference.
Owner Names

The MicroKernel allows you to restrict access to files by specifying an owner name. Because owner names are optional, the files you use with the utility may or may not require an owner name. Owner names are case-sensitive.

Conceptually, owner names are like passwords. They are not the same as user or group names, which you can set in PCC. For example, an owner name of “Master” is not the same as the default user Master.

A “short” owner name can be up to 8 bytes. A “long” owner name can be up to 24 bytes. Note, however, that once a long owner name is specified, the data file cannot be read by a database engine prior to PSQL v10.10. Also, a data file with a long owner name cannot be rebuilt to a file format prior to 9.5 unless the owner name is first removed. An owner name, long or short, with less than the maximum allowed bytes is padded with spaces to the maximum length (8 or 24 bytes).

With relational access, an ODBC error results if you attempt to manipulate a table that is restricted by an owner name. (For example in PCC, if you double-click the table name or attempt to delete the table.) You can supply owner names with the GRANT statement or the SET OWNER statement.

Use the GRANT statement to grant access to a particular user or group, and then manipulate the table via relational access through ODBC. The Master user must supply the GRANT statement with the correct owner name. See GRANT in SQL Engine Reference.

Use SET OWNER to specify one or more owner names to use during the current database connection. See SET OWNER in SQL Engine Reference.

Owner Names Tasks

Owner names can be set and cleared.

Setting or Clearing an Owner Name

You set an owner name to restrict access to a data file. Clear an owner name to remove the restriction.

Note You can also use a GRANT statement to supply an owner name. See Owner Name in SQL Engine Reference. PCC currently does not provide a way to specify an owner name through the security properties of a file.

To set or clear an owner name

1. Click Options > Set - Clear Owner in the menu bar. The Set - Clear Owner Name dialog appears.
2 In the **MicroKernel File** box, specify the file for which you want to set or clear an owner name. Then, to clear the owner name, click **Clear Owner** and specify the file's owner name in the **Current Owner** field.

3 To set the owner name, click **Set Owner**, specify the file's new owner name in the **New Owner** field, then select any desired options.

   - Select **Permit read-only access without an owner name** to allow all users read-only access to the data file.
   - Select **Encrypt data in file** to ensure that unauthorized users do not examine your data using a debugger or a file dump utility. Only select this option if data security is important to your environment because encryption and decryption require additional processing time.
   - Select **Long Owner Name** to create an owner name up to 24 bytes. (A “short” owner name can be up to 8 bytes.) Once a long owner name is specified, the data file cannot be read by a database engine prior to PSQL v10.10. Also, a data file with a long owner name cannot be rebuilt to a file format prior to 9.5 unless the owner name is first removed.

4 Click **Execute** to apply the options.
Statistics Report

Generating a statistics report is a good way to determine whether a file can be logged by the database engine's transaction durability feature. The report shows whether the file has system data and if a key is unique. (A unique key lacks the “D” flag, which indicates that duplicates are allowed.) The statistics report provides metadata about the file. This information can be used when you troubleshoot problems or to help you create similar files.

Statistics Report Tasks

The following task lists the steps to create a statistics report.

To create a statistics report for an existing data file

1. Click **Options > Create Stat Report** from the menu on the main window. The Maintenance utility displays the **Statistics Report** dialog box.

2. Specify a data file to use and a report file name. If you want to view the report when it is created, select the **View Report** check box.

   If you choose to view the report, the Maintenance utility displays the View File window shown next.

The informational headings in a status report correspond to the controls in the File Information Editor, which is described in **File Information Editor**.

The legend at the bottom of the statistics report explains the symbols used in the key/segment portion of the report. This information includes items such as the number of keys and key segments, the position of the key in the file, and the length of the key:

Legend:
Manipulating Btrieve Data Files with Maintenance

< = Descending Order
D = Duplicates Allowed
I = Case Insensitive
M = Modifiable
R = Repeat Duplicate
A = Any Segment (Manual)
L = All Segments (Null)
* = The values in this column are hexadecimal.
?? = Unknown
-- = Not Specified
Indexes

An index is a structure that sorts all the key values for a specific key. Btrieve access permits overlapping indexes (an index that includes a partial column). Relational access through ODBC does not permit overlapping indexes. (You can create an overlapping index with the File Information Editor, which you can display by clicking the Goto Editor button.)

Index Tasks

You perform the following tasks pertaining to indexes:
- Creating Indexes
- Dropping Indexes

Creating Indexes

You cannot create an index for a file unless the file has at least one key defined. You can create a key with the File Information Editor (see File Information Editor).

➢ To create an index

1. Click Index > Create from the main menu, which opens the Create Index dialog box.

2. Complete the following options in the Create Index dialog box.

<table>
<thead>
<tr>
<th>Index Type</th>
<th>Specify whether to create an internal or external index. Internal indexes are dynamically maintained as part of the data file. External indexes are separate files you generate as needed.</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>An external index file is a standard data file that contains records sorted by the key you specify. Each record consists of the following:</td>
</tr>
<tr>
<td></td>
<td>• A 4-byte address identifying the physical position of the record in the original data file</td>
</tr>
<tr>
<td></td>
<td>• A key value</td>
</tr>
<tr>
<td>Data File</td>
<td>Specify the name of the data file for which you want to create the index.</td>
</tr>
<tr>
<td>External Index File</td>
<td>Specify the name of the file to generate for an external index. Not applicable for internal indexes.</td>
</tr>
</tbody>
</table>
You can click Go To Editor to display the File Information Editor dialog box, which shows more complete information about the key. You can click Refresh Lists to read key information from the data file and refresh the Existing Key Numbers in Data File and Key Number to Use For Create lists. You must click Refresh Lists before you can create an index.

When you have completed the Create Index dialog box, click Execute to create the index. The amount of time required to create the index depends on how much data the file contains.

Dropping Indexes

Ensure that you understand the access performed by an application program before dropping an index. Certain functions fail (such as GET NEXT) if a required index is missing. This can result in an application program not functioning correctly.

To drop an index

1. Click Index > Drop from the main menu. The Drop Index dialog box appears.

2. Complete the following options in the Drop Index dialog box.

3. Click Refresh List to get the key information from the file you have specified.
Data

The commands in the Data menu allow you to import, export, and copy records in data files. You can also recover data after a system failure with the Roll Forward feature. See Roll Forward Command for a discussion of Roll Forward.

Importing and Exporting ASCII File Format

When you save data, records in the ASCII file have the following format. You can use an ASCII text editor to create files that you can load, as long as they adhere to these specifications. Note that most text editors do not support editing binary data.

- The first field is a left-adjusted integer (in ASCII) that specifies the length of the record. (When calculating this value, ignore the carriage return/line feed that terminates each line.) The value in this first field matches the record length specified in the data file.
  - For files with fixed-length records, the length you specify should equal the record length of the data file.
  - For files with variable-length records, the length you specify must be at least as long as the fixed-record length of the data file.
- A separator (a comma or a blank) follows the length field.
- The record data follows the separator. The length of the data is the exact number of bytes specified by the length field. If you are creating an import ASCII file using a text editor, pad each record with blank spaces as necessary to fill the record to the appropriate length.
- A carriage return/line feed (0D0A hexadecimal) terminates each line. The Maintenance utility does not insert the carriage return/line feed into the data file.
- The last line in the file must be the end-of-file character (CTRL+Z or 1A hexadecimal). Most text editors automatically insert this character at the end of a file.

Figure 31 shows the correct format for records in the input ASCII file. For this example, the data file has a defined record length of 40 bytes.

Figure 31    Format for Records in Input Sequential Files

Data Tasks

You can perform the following data tasks with the Maintenance utility:

- To import ASCII data
- To export ASCII records
To copy records between MicroKernel data files
To recover (Roll Forward) changes made to a data file between the time of the last backup and a system failure, see the Logging, Backup, and Restore chapter.

Importing Records From an ASCII File
You can use the Maintenance utility to import records from an ASCII file to a standard data file. This operation does not perform any conversions on the data. You can create an import file using a text editor or the Maintenance utility (see Exporting Records to an ASCII File).

➢ To import ASCII data
1. Click Data > Load from the main menu. The Load dialog box appears.

   Figure 32  Load Dialog Box

The ASCII file you specify must adhere to the specifications explained in Importing and Exporting ASCII File Format. The record length of the standard data file you specify must be compatible with the records in the ASCII file.

2. Click Execute to import the records.

   While importing data, the Maintenance utility shows the number of records being imported, the percentage of records imported, and a status message. You can continue working in the Maintenance utility (for example, you can open another Load dialog box).

Exporting Records to an ASCII File
You can use the Maintenance utility to export records from a data file to an ASCII file.

➢ To export ASCII records
1. Click Data > Save from the main menu. The Save Data dialog box appears.
2 In the **Save Data** dialog box, specify the following options.

<table>
<thead>
<tr>
<th><strong>From MicroKernel File</strong></th>
<th>Specifies the name of the existing MicroKernel-compatible file you want to save.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>To Sequential File</strong></td>
<td>Specifies the name of the sequential file to create.</td>
</tr>
<tr>
<td><strong>Use An Index</strong></td>
<td>Uses a specified index when sorting the records for export. By default, the Maintenance utility does not use an index, meaning that records are exported according to their physical position in the data file.</td>
</tr>
<tr>
<td><strong>Internal Index #</strong></td>
<td>Uses the specified key number. Click <strong>Refresh Index List</strong> to update the available indexes if you change file in the <strong>From MicroKernel File</strong> box.</td>
</tr>
<tr>
<td><strong>External Index File</strong></td>
<td>Uses the specified external index. (To create an external index, refer to <a href="#">Creating Indexes</a>.)</td>
</tr>
<tr>
<td><strong>Direction</strong></td>
<td><strong>Forward</strong>: This is the default setting and indicates the utility recovers the file from the beginning. <strong>Backward</strong>: This option recovers data from the end of the file. <strong>Forward and Backward</strong>: This option reads the file forward until it fails. Then it starts at the end of the file and reads the file backward until it reaches the record that failed previously or encounters another failure. <strong>Backward and Forward</strong>: Indicates the utility reads the file backward until it fails. Then it starts at the beginning of the file and reads the file forward until it reaches the record that failed previously or encounters another failure.</td>
</tr>
</tbody>
</table>

3 Click **Execute** to export the data. The Maintenance utility creates the specified ASCII file using the format described in [Importing and Exporting ASCII File Format](#). You can then edit the ASCII file and use the **Load** command to import the edited text to another standard data file.
Manipulating Btrieve Data Files with Maintenance

Copying Records Between Data Files
You can use the Maintenance utility to copy data from one MicroKernel data file to another. The record lengths for both data files you specify must be the same.

To copy records between MicroKernel data files
1. Click Data > Copy from the main menu. The Copy Data dialog box appears.

   Figure 34   Copy Data Dialog Box

2. Enter the name of the file you want to copy in the From MicroKernel File box and then specify the path where you want to copy the file in the To MicroKernel File box.

   The record lengths for both data files you specify must be the same.

Recovering (Roll Forward) Changes to a Data File
See the Logging, Backup, and Restore.
Btrieve Command-Line Maintenance Utility (butil)

Use this utility if you prefer a command-line interface or if you want to start or stop continuous operation. The Btrieve Maintenance utility is also available in a command-line format that runs on the server (from a DOS command prompt on Windows platforms) or locally on DOS, Linux and Windows clients. You can execute maintenance utility commands from the command line or through a command file you create. Before you perform commands in the Btrieve Maintenance utility, also called butil, it is important you understand some concepts and elements addressed in the Commands.

The Btrieve Command-Line Maintenance utility performs the following file and data manipulations:

- Importing and Exporting Data
- Creating and Modifying Data Files
- Viewing Data File Statistics
- Displaying MicroKernel Engine Version
- Unloading the MicroKernel Engine and Requester (DOS only)
- Performing Continuous Operations

Return Codes

When butil finishes executing, it returns an exit code or DOS “errorlevel” return code to the operating system. The return codes are as follows:

<table>
<thead>
<tr>
<th>Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUCCESS_E = 0</td>
<td>Requested operation succeeded.</td>
</tr>
<tr>
<td>PARTIAL_E = 1</td>
<td>Requested operation completed, but with errors.</td>
</tr>
<tr>
<td>INCOMPLETE_E = 2</td>
<td>Requested operation did not complete.</td>
</tr>
<tr>
<td>USAGE_E = 3</td>
<td>Syntax error in input, display usage screen and exit.</td>
</tr>
</tbody>
</table>

Commands

The following table lists the commands that you can use with the Command-line Maintenance Utility.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clone</td>
<td>Creates a new, empty data file using an existing file’s specifications.</td>
</tr>
<tr>
<td>Clrown</td>
<td>Clears the owner name of a data file.</td>
</tr>
<tr>
<td>Copy</td>
<td>Copies the contents of one data file to another.</td>
</tr>
<tr>
<td>Create</td>
<td>Creates a data file.</td>
</tr>
<tr>
<td>Drop</td>
<td>Drops an index.</td>
</tr>
<tr>
<td>Endbu</td>
<td>Ends continuous operation on data files defined for backup.</td>
</tr>
<tr>
<td>Index</td>
<td>Creates an external index file.</td>
</tr>
</tbody>
</table>
Manipulating Btrieve Data Files with Maintenance

Viewing Command Usage Syntax
To view a summary of each command usage, enter the **butil** command at the file server.

**Command Format**
The format for the Maintenance utility command line is as follows:

```
butil [-command [parameter ...]] | @commandFile
```

- `-command` A Maintenance utility command, such as COPY. You must precede the command with a dash (-), and you must enter a space before the dash. Table 77 lists the commands.
- `parameter` Information that the command may require. Discussions of the individual commands provide details when applicable.
- `@commandFile` Fully qualified file name of a command file.

**Command Files**
You can use a command file to do the following:

- Execute a command that is too long to fit on the command line.
- Execute a command that you use often (by entering the command once in the command file and then executing the command file as often as you want).
- Execute a command and write the output to a file, using the following command format:
  
  `butil @CommandFile [commandOutputFile]`

  For each command executed, the resulting output file shows the command followed by its results. All messages appear on the server console screen, as well.
- Execute multiple commands sequentially.

**Table 77 Command-Line Maintenance Utility Commands**

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Load</td>
<td>Loads the contents of an unformatted file into a data file.</td>
</tr>
<tr>
<td>Recover</td>
<td>Reads data sequentially from a data file and writes the results to an unformatted file. (The DOS version does not support ROLLFWD.) Use this command if you have a damaged file.</td>
</tr>
<tr>
<td>Rollfwd</td>
<td>Recovers changes made to a data file between the time of the last backup and a system failure. See Performing Archival Logging.</td>
</tr>
<tr>
<td>Save</td>
<td>Reads data along a key path and writes the results to a sequential file.</td>
</tr>
<tr>
<td>Setowner</td>
<td>Assigns an owner name to a data file.</td>
</tr>
<tr>
<td>Sindex</td>
<td>Creates an index.</td>
</tr>
<tr>
<td>Startbu</td>
<td>Starts continuous operation on files defined for backup. See the chapter Logging, Backup, and Restore.</td>
</tr>
<tr>
<td>Stat</td>
<td>Reports statistics about file attributes and current sizes of data files.</td>
</tr>
<tr>
<td>Stop (DOS only)</td>
<td>Unloads the MicroKernel Engine and Requester.</td>
</tr>
<tr>
<td>Ver</td>
<td>Displays the version of the database engine and requester that is loaded at the server.</td>
</tr>
</tbody>
</table>

---

292
Command files contain the same information as that required on the command line.

**Rules for Command Files**

Observe the following rules when creating a Maintenance utility command file:

- You cannot split a single parameter across two lines.
- You must end each command with `<end>` or `[end]`. You must also end each command with `<end>` when trying to execute multiple commands. The `<end>` or `[end]` must be lowercase.

**Command File Example**

The following is an example command file, COPYCRS.CMD. The file calls the **BUTIL -CLONE** command to create the NEWCRS.MKD file by cloning the COURSE.MKD file, and the **-CREATE** command to create the NEWFILE.DTA file by using the description provided in the NEWFILES.DES description file.

```
-clone newcrs.mkd course.mkd <end>
-create newfile.dta newfiles.des <end>
```

The following command uses the COPYPATS.CMD file and writes the output to the COPYPATS.OUT file:

```
butil @copypats.cmd copypats.out
```

**Description Files**

Description files are ASCII text files that contain descriptions of file and key specifications that the Maintenance utility can use to create data files and indexes. Some users employ description files as a vehicle for archiving information about the data files they have created. For more information about the description file format, see [Description Files](#).

**Extended File Support**

The size of the database engine data file can be larger than the operating system file size limit. When you export data from an extended MicroKernel file to an unformatted file, the size of the unformatted file can exceed the database engine file size limit because of the differences in the physical format.

When you are exporting large files, the Interactive Maintenance utility detects that the unformatted file has exceeded the operating system file size limit (2 GB) and starts creating extension files. This process is transparent. Extension files and the original unformatted file must reside on the same volume. The extension file uses a naming scheme in which the filenames are similar to the base filename. In contrast to native MicroKernel extension files which use a caret “^” to indicate extension file status, the unformatted extension files use a tilde “~” to avoid overwriting any existing extended MicroKernel Engine files with the same base file name. The first export extension file is the same base file name with “~01” extension. The second extension file is “~02,” and so on. These extensions are appended in hexadecimal format.

While the naming convention supports up to 255 extension files, the current maximum number of extension files is 64, thus supporting files as large as 128 GB.

To Save or Recover huge files to unformatted files, see the respective command. Also, when you import data from an unformatted file, the utility detects if the file has extensions and loads the data from the extension file.
**Owner Names**

The MicroKernel allows you to restrict access to files by specifying an owner name. Because owner names are optional, the files you use with the utility may or may not require an owner name.

A “short” owner name can be up to 8 bytes. A “long” owner name can be up to 24 bytes. Once a long owner name is specified, the data file cannot be read by a database engine prior to PSQL v10.10. Also, a data file with a long owner name cannot be rebuilt to a file format prior to 9.5 unless the owner name is first removed. An owner name, long or short, with less than the maximum allowed bytes is padded with spaces to the maximum length (8 or 24 bytes).

If the file requires an owner name, you must specify it using the `/O` option. You can specify one of the following:

- Single owner name.
- List of up to eight owner names. Separate the owner names with commas.
- Asterisk (*). The utility prompts you for the owner name. With the `rollfwd` command, the utility prompts you for a list of owner names separated by commas.

Owner names are case-sensitive. If you enter owner names on the command line, the utility discards leading blanks. If you specify an asterisk, the utility does not discard leading blanks.

**Redirecting Error Messages**

Be sure that you specify a fully qualified file name (including a drive letter or UNC path) when redirecting error messages.

- To redirect error messages to a file

  Use the following command format.

  ```
  buil -command commandParameters > filePath
  ```

**ASCII File Format**

See Importing and Exporting ASCII File Format in the Interactive Maintenance utility section.

**Rules for Specifying File Names on Different Platforms**

When you run `butil` on a Windows-based platform or a Linux-based platform, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.
Importing and Exporting Data

This section provides detailed information on importing and exporting data using the following butil commands: **Copy**, **Load**, **Recover**, and **Save**.

Table 78  Commands to Import and Export Data

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy</td>
<td>Copies the contents of one data file to another.</td>
</tr>
<tr>
<td>Load</td>
<td>Loads the contents of a sequential file into a data file.</td>
</tr>
<tr>
<td>Recover</td>
<td>Reads data sequentially from a data file and writes the results to a sequential file.</td>
</tr>
<tr>
<td>Save</td>
<td>Reads data along a key path and writes the results to a sequential file.</td>
</tr>
</tbody>
</table>

**Copy**

The Copy command copies the contents of one MicroKernel file to another. Copy retrieves each record in the input data file and inserts it into the output data file. The record size must be the same in both files. After copying the records, Copy displays the total number of records inserted into the new data file.

Note Copy performs in a single step the same function as a Recover command followed by a Load command.

Using the Copy command, you can create a data file that contains data from an old file, but has new key characteristics.

➢ To copy a MicroKernel data file

1. Use the Create command to create an empty data file with the desired key characteristics (key position, key length, or duplicate key values).
   
   or
   
   Use Clone to create an empty data file using the characteristics of an existing file.

2. Use the Copy command to copy the contents of the existing data file into the newly created data file.

Format

butil -copy sourceFile outputFile [/O< owner1 | *>  
[<O<owner2 | *>]] [/[UID]name /PWDpword [/DBdbname]]

sourceFile  The fully qualified name of the data file from which to transfer records. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

outputFile The fully qualified name of the data file into which to insert records. The output data file can contain data or be empty. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner1  The owner name of the source data file, if required. If only the output data file requires an owner name, specify /O followed by a blank for owner1 (as illustrated in the example).
Example

The following command copies the records in COURSE.MKD to NEWCRS.MKD. The COURSE.MKD input file does not require an owner name, but the NEWCRS.MKD output file uses the owner name Pam.

\texttt{butil -copy course.mkd newcrs.mkd /O /OPam}

If you omit the first /O from this example, the utility assumes that the owner name Pam belongs to the input data file, not the output data file.

Load

The \texttt{load} command inserts records from an input ASCII file into a file. The input ASCII file can be a single file or an extended file (the base file plus several extension files). \texttt{Load} performs no conversion on the data in the input ASCII file. After the utility transfers the records to the data file, it displays the total number of records loaded.

\textbf{Note} The \texttt{load} command opens the output file in Accelerated mode; during a load operation, the database engine does not log the file. If you are using archival logging, back up your data files again after using the \texttt{load} command.

Extended files: If the utility finds the next extension file, it continues the load process. Do not delete any extension file created earlier by the \texttt{save} and \texttt{recover} commands. If the file has three extensions and the user deletes the second one, \texttt{load} stops loading records after processing the first extension file.

If \texttt{save} or \texttt{recover} created three extension files and a fourth one exists from a previous \texttt{save} or \texttt{recover}, \texttt{load} reads the records from the fourth extension and inserts them into the database engine file. If a fourth file exists, then you need to delete it before starting the \texttt{load} process.

Before running the \texttt{load} command, you must create the input ASCII file and the data file. You can create the input ASCII file using a standard text editor or an application; the input ASCII file must have the required file format (see \texttt{Importing and Exporting ASCII File Format}). You can create the data file using either the \texttt{Create} or the \texttt{Clone} command.
Importing and Exporting Data

Format

```
butil -load unformattedFile outputFile [/O<owner |*>] [/UID<name> /PWD<pword> [/DB<dbname>]]
```

- `unformattedFile`: The fully qualified name of the ASCII file containing the records to load into a data file. For Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

- `outputFile`: The fully qualified name of the data file into which to insert the records. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

- `/O<owner>`: The owner name for the data file, if required.

- `/UID<name>`: Specifies the name of the user authorized to access a database with security enabled.

- `/PWD<pword>`: Specifies the password for the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.

- `/DB<dbname>`: Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Example

The following example loads sequential records from the `COURSE.TXT` file into the `COURSE.MKD` file. The owner name of the `COURSE.MKD` file is Sandy.

```
butil -load course.txt course.mkd /OSandy
```

Recover

The `recover` command extracts data from a MicroKernel file and places it in an ASCII file that has the same format as the input ASCII file that the `load` command uses. This is often useful for extracting some or all of the data from a damaged MicroKernel file. The `recover` command may be able to retrieve many, if not all, of the file's records. You can then use the `load` command to insert the recovered records into a new, undamaged MicroKernel file.

Note

The maintenance utility performs no conversion on the data in the records. Therefore, if you use a text editor to modify an output file containing binary data, be aware that some text editors may change the binary data, causing the results to be unpredictable.

Format

```
```

- `sourceFile`: The fully qualified name of the data file from which to recover data. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

- `unformattedFile`: The fully qualified name of the ASCII file where the utility should store the recovered records.

- `/O<owner>`: The owner name for the data file, if required.
Manipulating Btrieve Data Files with Maintenance

For each record in the source file, if the `recover` command receives a variable page error (Status Code 54), it places all the data it can obtain from the current record in the unformatted file and continues the recovery process.

The utility produces the following messages:
- informs you about the name of the last extension file created
- checks if the next extension file exists, and if so, tells you to delete it
- if you move the extended unformatted files to a different location, you are prompted to move the base file and all of its extension files

Example

The following example extracts records from the COURSE.MKD file and writes them into the COURSE.TXT file.

```
butil -recover course.mkd course.txt
```

Save

The `save` command retrieves records from a MicroKernel file using a specified index path and places them in an ASCII file that is compatible with the required format for the `load` command. You can then edit the ASCII file and use the `load` command to store the edited data in another data file. (See Importing and Exporting ASCII File Format for more information about the ASCII file format.)

`save` generates a single record in the output ASCII file for each record in the input data file. Upon completion, `save` displays the total number of records saved.
Note: The Maintenance utility performs no conversion on the data in the records. Therefore, if you use a text editor to modify an output file containing binary data, be aware that some text editors may change the binary data, causing the results to be unpredictable.

**Format**

```
butil -save sourceFile unformattedFile [Y indexFile | N <keyNumber | -1>] [O<owner1 | *> [/O<owner2 | *>]] [Q] [J] [/I] [/UIDuname /PWDpword /DBdbname]
```

- `sourceFile`: The fully qualified name of the data file containing the records to save. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.
- `unformattedFile`: The fully qualified name of the ASCII file where you want the utility to store the records.
- `indexFile`: The fully qualified name of an external index file by which to save records if you do not want to save records using the default of the lowest key number.
- `keyNumber`: The key number (other than 0) by which to save records if you do not want to save records using the default of the lowest key number.
- `-1`: The specification for saving the records in physical order using the Btrieve Step operations.
- `/Oowner1`: The owner name for the source file, if required. If only the index file requires an owner name, specify `/O` followed by a blank for `owner1`.
- `/Oowner2`: The owner name for the index file, if required.
- `/Q`: Indicates whether to replace an existing unformatted file. By default, the Maintenance utility overwrites the existing files. If you specify this option and a file with the same name exists, the utility returns an error message.
- `/J`: Indicates BACKWARD reading of the file. If you specify this option, the utility recovers data from the database engine file using GET LAST and PREVIOUS operations. The default is forward reading, using GET FIRST and NEXT operations.
- `/I`: Indicates FORWARD reading of the file. Although the default is forward reading, you can use this option to indicate FORWARD and BACKWARD reading. This means that if you specify both `/I` and `/J`, respectively, the utility reads the file forward until it fails. Then it starts at the end of the file and reads backwards until it reaches the record that failed previously or encounters another failure.

If you specify `/I` first, the utility reads backwards and then reads forward.

- `/UID<name>`: Specifies the name of the user authorized to access a database with security enabled.
- `/UIDuname`:
- `/PWD<word>`: Specifies the password for the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.
- `/PWDpword`:
- `/DB<name>`: Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.
Manipulating Btrieve Data Files with Maintenance

The utility produces the following messages:

- informs you about the name of the last extension file created
- checks if the next extension file exists, and if so, tells you to delete it
- if you move the extended unformatted files to a different location, you are prompted to move the base file and all of its extension files

Examples

The following two examples illustrate how to use the SAVE command to retrieve records from a data file.

This example uses a NEWCRS.IDX external index file to retrieve records from the COURSE.MKD file and store them in an unformatted text file called COURSE.TXT:

```
butil save course.mkd course.txt newcrs.idx
```

The following example retrieves records from the COURSE.MKD file using key number 3 and stores them in an unformatted text file called COURSE.TXT:

```
butil -save course.mkd course.txt n 3
```
Creating and Modifying Data Files

This section includes detailed information on creating and modifying data files using the following BUTIL commands: **Clone**, **Clowner**, **Create**, **Drop**, **Index**, **Setowner**, and **Sindex**. This section also includes information about removing unused space in a Btrieve data file, which is discussed in **Compacting Btrieve Data Files**.

**Caution** No two files can share the same file name and differ only in their file name extension if both files are in the same directory. For example, do not name a data file Invoice.btr and another one Invoice.mkd in the same directory. This restriction applies because the database engine uses the file name for various areas of functionality while ignoring the file name extension. Since only the file name is used to differentiate files, files that differ only in their file name extension look identical to the database engine.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Clone</strong></td>
<td>Creates a new, empty data file using an existing file’s specifications.</td>
</tr>
<tr>
<td><strong>Clowner</strong></td>
<td>Clears the owner name of a data file.</td>
</tr>
<tr>
<td><strong>Create</strong></td>
<td>Creates a data file.</td>
</tr>
<tr>
<td><strong>Drop</strong></td>
<td>Drops an index.</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>Creates an external index file.</td>
</tr>
<tr>
<td><strong>Setowner</strong></td>
<td>Assigns an owner name to a data file.</td>
</tr>
<tr>
<td><strong>Sindex</strong></td>
<td>Creates an index.</td>
</tr>
</tbody>
</table>

**Clone**

The **clone** command creates a new, empty file with the same file specifications as an existing file (including any supplemental indexes, but excluding the owner name). The new data file includes all the defined key characteristics (such as key position, key length, or duplicate key values) contained in the existing file.

The **clone** command ignores all MicroKernel configuration options that affect file statistics (such as **System Data**) except file version. The **clone** command creates a new file using the database engine file version you specify with the **Create File Version** option.
Format


outputFile The fully qualified file name to use for the new, empty data file. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

sourceFile The fully qualified file name of the existing data file to replicate. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

/Oowner The owner name, if any, for the source data file. The new data file does not have an owner name. See Owner Names for more information.

/pagecompresson Turns on page compression for outputFile provided the following conditions are true:
• The version of the PSQL database engine is PSQL 9.5 or newer.
• The setting for Create File Version is 0950 (9.5) or higher. See Create File Version.

/pagecompressoff Turns off page compression for outputFile. This parameter has no effect if sourceFile does not contain page compression.

/recordcompresson Turns on record compression for outputFile.

/recordcompressoff Turns off record compression for outputFile. This parameter has no effect if sourceFile does not contain record compression.

/UID<name> Specifies the name of the user authorized to access a database with security enabled.

/UIuname

/PWD<word> Specifies the password for the user who is identified by uname. Pword must be supplied if uname is specified.

/PWDpword

/DB<name> Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

/DBdbname

Remarks

Btrieve 6.0 and later allows a maximum of 23 key segments in a data file with a page size of 1,024 bytes. Therefore, the CLONE command sets the page size in the new data file to 2,048 bytes if the existing data file contains 24 key segments and has a page size of 1,024 bytes. This occurs if the existing data file has a format earlier than 6.0 and the database engine was not loaded with the Create File Version option set to 5.x or 6.x.

If you are cloning a pre-7.x file, ensure that the database engine is configured to create the file format version that you want the new file to be. For example, if you want to clone a 6.15 file in 9.5 format, ensure that the Microkernel File Format Version option is set to 9.5.

Note If your source file is in 8.x format or later and it does not contain system data, your output file will not contain system data, regardless of the database engine configuration. To add system data to an existing file, refer to Getting Started With PSQL.
If you are trying to recover from receiving Status Code 30 (The file specified is not a MicroKernel file) and you suspect that the header page of the source file might be damaged, try creating the new MicroKernel file using the \textbf{Create} command with a description file.

\textbf{Example}

The following command creates the NEWCRS.MKD file by cloning the COURSE.MKD file.

\begin{verbatim}
butil -clone newcrs.mkd course.mkd
\end{verbatim}

\textbf{Clrowner}

The \textbf{clrowner} command clears the owner name of a MicroKernel file.

\textbf{Format}

\begin{verbatim}
butil -clrowner sourceFile /O<owner | *> [/UID username /PWD pword [/DB dbname]]
\end{verbatim}

- \texttt{sourceFile} - The fully qualified file name of the data file. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.
- \texttt{/O<owner | *>} - The owner name to clear. See Owner Names for more information.
- \texttt{/UID username} - Specifies the name of the user authorized to access a database with security enabled.
- \texttt{/PWD pword} - Specifies the password for the user who is identified by \textit{uname}. \textit{Pword} must be supplied if \textit{uname} is specified.
- \texttt{/DB dbname} - Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

\textbf{Example}

The following command clears the owner name for the TUITION.MKD file. The owner name for the file is Sandy.

\begin{verbatim}
butil -clrowner tuition.mkd /OSandy
\end{verbatim}

\textbf{Create}

The \textbf{create} command generates an empty MicroKernel file using the characteristics you specify in a description file. Before you can use the \textbf{create} command, you must create a description file to specify the new key characteristics. For more information, see Description Files.

\textbf{Format}

\begin{verbatim}
butil -create outputFile descriptionFile [< Y | N >] [/UID username /PWD pword [/DB dbname]]
\end{verbatim}

- \texttt{outputFile} - The fully qualified file name of the database engine file to create. If the file name is the name of an existing MicroKernel file, this command creates a new, empty file in place of the existing file. Any data that was stored in the existing file is lost and cannot be recovered. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.
- \texttt{descriptionFile} - The fully qualified name of the description file containing the specifications for the new MicroKernel file.
Manipulating Btrieve Data Files with Maintenance

Example
The following command creates a file named COURSE.MKD using the description provided in the CREATE.DES description file.

```
butil -create course.mkd create.des
```

Sample Description File for the CREATE Command
The sample description file shown in Figure 35 creates a MicroKernel formatted file. The file is specified to have a page size of 512 bytes and 2 keys. The fixed-length portion of each record in the file is set to 98 bytes. The file specifies variable-length records with no blank truncation, record compression, and variable-tail allocation tables (VATs). The free space threshold is set to 20 percent. Allocation is set to 100 pages. The MicroKernel preallocates 100 pages, or 51,200 bytes, when it creates the file.

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y

key=0 position=1 length=5 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=y

key=0 position=6 length=10 duplicates=y
modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=n

key=0 position=16 length=2 duplicates=n
modifiable=y type=numeric descending=y
nullkey=n segment=n

name=c:\myacsfiles\upper.alt
```

Key 0 is a segmented key with two duplicatable, nonmodifiable string segments and a null value of 20 hexadecimal (space) specified for both segments. Key 0 uses the collating sequence upper.alt.

Key 1 is a numeric, nonsegmented key that does not allow duplicates but permits modification. It is sorted in descending order.
Creating and Modifying Data Files

Drop

The drop command removes an index from a file and adjusts the key numbers of any remaining indexes, subtracting 1 from each subsequent key number. If you do not want to renumber the keys, you can add 128 to the key number you specify to be dropped. This renumbering feature is available only for 6.0 and later files.

Format

butil -drop sourceFile < keyNumber | SYSKEY >

[sourceFile | SYSKEY>

sourceFile The fully qualified name of the file from which you are dropping the index. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

keyNumber The number of the key to remove. To preserve the original key numbers, add a 128 bias to the key number you specify.

SYSKEY Instructs the utility to drop the system-defined log key (also called system data). Dropping the system-defined log key does not delete values from the records; the database engine still assigns unique system-defined log key values to newly inserted records.

However, the database engine cannot perform logging for a file from which the system-defined log key is dropped, if no user-defined unique keys exist. For this reason, you should use this option only if you suspect that the system-defined log key is corrupt and you intend to readd it.

The sindex command allows you to reuse the system-defined log key once you have dropped it.

/O owner The owner name for the file, if required.

/UID <name> Specifies the name of the user authorized to access a database with security enabled.

/PWD <word> Specifies the password for the user who is identified by name. Pword must be supplied if name is specified.

/DB <name> Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Examples

In both of the following examples, COURSE.MKD has three keys. The original keys in the file are numbered 0, 1, and 2.

In the first example, the butil -drop command drops key number 1 from the COURSE.MKD file and renumbers the remaining key numbers as 0 and 1.

butil -drop course.mkd 1

In the following example, the butil -drop command drops key number 1, but does not renumber the keys. The key numbers remain 0 and 2.

butil -drop course.mkd 129
Index

The `index` command builds an external index file for an existing MicroKernel file, based on a field not previously specified as a key in the existing file. Before you can use the `index` command, you must create a description file to specify the new key characteristics. For more information about description files, see Description Files.

The records in the new file consist of the following:

- The 4-byte address of each record in the existing data file.
- The new key value on which to sort.

\[\text{Note}\] If the key length you specify in the description file is 10 bytes, the record length of the external index file is 14 bytes (10 plus the 4-byte address).

Format

```
butil -index sourceFile indexFile descriptionFile [/O<owner | *>] [/UID<name> /PWD<word> /DB<dbname>]
```

- `sourceFile`: The fully qualified name of the existing file for which to build an external index. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

- `indexFile`: The fully qualified name of the index file in which the database engine should store the external index.

- `descriptionFile`: The fully qualified name of the description file you have created containing the new key definition. The description file should contain a definition for each segment of the new key.

- `/O<owner>`: The owner name for the data file, if required.

- `/UID<name>`: Specifies the name of the user authorized to access a database with security enabled.

- `/PWD<word>`: Specifies the password for the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.

- `/DB<name>`: Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.

Remarks

The `index` command creates the external index file and then displays the number of records that were indexed. To retrieve the data file's records using the external index file, use the `Save` command.

Sample Description File for the INDEX Command

The description file shown in the following illustration defines a new key with one segment. The key begins at byte 30 of the record and is 10 bytes long. It enables duplicates, is modifiable, is a STRING type, and uses no alternate collating sequence.
Creating and Modifying Data Files

Figure 36  Sample Description File for INDEX Command

```
position=30 length=10 duplicates=y modifiable=y
type=string alternate=n segment=n
```

Example

The following command creates an external index file called NEWCRS.IDX using a data file called COURSE.MKD. The COURSE.MKD file does not require an owner name. The description file containing the definition for the new key is called NEWCRS.DES.

```
butil -index course.mkd newcrs.idx newcrs.des
```

Setowner

The `setowner` command sets an owner name for a data file.

Format

```
butil -setowner sourceFile /O<owner> | *> level [/L] [/UIDuname /PWDpword [/DBdbname]]
```

- `sourceFile` The fully qualified name of the data file. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.
- `/O<owner>` The owner name to be set
- `level` The type of access restriction for the data file. The possible values for this parameter are as follows
  - 0: Requires an owner name for any access mode (no data encryption)
  - 1: Permits read access without an owner name (no data encryption)
  - 2: Requires an owner name for any access mode (with data encryption)
  - 3: Permits read access without an owner name (with data encryption)
- `/L` Designates a long owner name.
  - Owner names are case sensitive and can be short or long. A “short” owner name can be up to 8 bytes long. A “long” owner name can be up to 24 bytes long. For restrictions pertaining to long owner names, see the section Procedure in Btrieve API Guide for Set Owner (29).
- `/UID<name>` Specifies the name of the user authorized to access a database with security enabled.
- `/UIDuname` Specifies the name of the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.
- `/PWD<pword>` Specifies the password for the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.
- `/DB<name>` Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.
Examples
The following example creates a short owner for the course.mkd data file. The owner name is Sandy, and the restriction level is 1.

butil -setowner course.mkd /OSandy 1

The following example creates a long owner name for the billing.mkd data file, encrypts the owner name and file, and restricts all access modes.

butil -setowner billing.mkd /Ohr#Admin$945k7YY%svr 2 /L

Sindex
The sindex command creates an additional index for an existing MicroKernel file. By default, the key number of the new index is one higher than the previous highest key number for the data file, or you can instruct the database engine to use a specific key number. An exception is if a drop command previously removed an index without renumbering the remaining keys, thus producing an unused key number; in this case, the new index receives the first unused number.

You can instruct the database engine to use a specific key number for the new index with the key number option. The key number you specify must be a valid key number that is not yet used in the file. If you specify an invalid key number, you receive Status Code 6.

If you do not use the SYSKEY option with this command, you must create a description file that defines key specifications for the index before you can use the sindex command. For more information about description files, see Description Files.

Format
butil -sindex sourceFile <descriptionFile | SYSKEY> [keyNumber] [/O<owner | *>] [/UID<name> /PWD<word> /DB<dbname>]

Examples
The following example adds an index to the COURSE.MKD file. The name of the description file is NEWIDX.DES.

sourceFile The fully qualified name of the existing file for which to build an external index. When you run BUTIL for Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

descriptionFile The fully qualified name of the description file you have created containing the new key definition. The description file should contain a definition for each segment of the new key.

SYSKEY Instructs the utility to readd the system key on a file in which the system key was dropped.

/Oowner The owner name for the data file, if required.

/UID<name> Specifies the name of the user authorized to access a database with security enabled.

/PWD<word> Specifies the password for the user who is identified by uname. Pword must be supplied if uname is specified.

/DB<dbname> Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.
butil -sindex course.mkd newidx.des

The following example adds the system-defined key to the COURSE.MKD file. The system-defined key was dropped.

butil -sindex course.mkd syskey

**Compacting Btrieve Data Files**

You can use several commands in the BUTIL (Clone, Recover, and Load, respectively) to remove unused space in a data file to decrease its size.

➢ **To compact a Btrieve data file**

1. Rename your data file and then use the Clone option to create a blank data file using the original file name.

2. Use Recover to save the data from the clone file to an unformatted text file in sequential order.

3. Use Load to load the recovered data into the clone.

   Every record containing data will load into the newly created data file without blank records. (You can also perform this operation in the Btrieve Interactive Maintenance utility.)
**Viewing Data File Statistics**

This section includes information about generating a report that contains a data file’s characteristics and statistics using STAT.

**Stat**

The `stat` command generates a report that contains defined characteristics of a data file and statistics about the file’s contents. Using the `stat` command is a good way to determine if a file can be logged by the database engine’s transaction durability feature. The `stat` command reports indexes the same whether they were created by the Create Supplemental Index operation (in Btrieve 6.0 and later) or the Create operation.

**Format**

```
butil -stat <sourceFile> [/O<owner | *>] [/O<owner | *>] [/UIDuname /PWDpword [/DBdbname]]
```

- **sourceFile**: The fully qualified name of the data file for which to report statistics. For Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.
- **/O<owner>**: The owner name for the data file, if required.
- **/UID<name>**: Specifies the name of the user authorized to access a database with security enabled.
- **/UIDuname**:
- **/PWD<word>**: Specifies the password for the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.
- **/DB<name>**: Specifies the name of the database on which security is enabled. If omitted, the default database is assumed.
- **/DBdbname**:

**Example**

The following example reports file statistics for the PATIENTS.DTA file. The data file does not have an owner name.

```
butil -stat patients.dta
```

The following example shows the resulting report:

```
**************************************************
File Statistics for PATIENTS.DTA

File Version = 8.00
Page Size = 2048
Page Preallocation = No
Key Only = No
Extended = No

Total Number of Records = 16
Record Length = 104
Record Compression = No
Variable Records = No

Available Linked Duplicate Keys = 0
```

310
Balanced Key = No  
Log Key = 1  
System Data = No  
Total Number of Keys = 3  
Total Number of Segments = 4

<table>
<thead>
<tr>
<th>Key</th>
<th>Segment</th>
<th>Position</th>
<th>Length</th>
<th>Type</th>
<th>Flags</th>
<th>Null Values*</th>
<th>Unique</th>
<th>ACS Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>21</td>
<td>20</td>
<td>String</td>
<td>MD</td>
<td>16</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>7</td>
<td>12</td>
<td>String</td>
<td>MD</td>
<td>16</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>String</td>
<td>M</td>
<td>16</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>83</td>
<td>10</td>
<td>String</td>
<td>MD</td>
<td>7</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Alternate Collating Sequence (ACS) List:  
0 UPPER  

Legend:  
< = Descending Order  
D = Duplicates Allowed  
I = Case Insensitive  
M = Modifiable  
R = Repeat Duplicate  
A = Any Segment (Manual)  
L = All Segments (Null)  
* = The values in this column are hexadecimal.  
?? = Unknown  
-- = Not Specified

This example shows that the file called PATIENTS.DTA is an 8.0 file. (The version number indicates the earliest Btrieve version that can read the file format.) The file has a page size of 2,048 bytes and has no preallocated pages. This is not a key-only file, nor is it an extended file.

Sixteen records have been inserted into the file. The file was defined with a record length of 104 bytes, does not use record compression, and does not allow variable-length records.

There are no linked duplicate keys available in the file. The file does not use balanced indexing. The MicroKernel performs logging using Key 1, and the file contains no system-defined data. The file has three keys comprised of four key segments.

Note Indexes created with Sindex are designated with the letter R by default unless you specified the Reserved Duplicate Pointer element.

The STAT report also provides information about specific keys. For example, the report shows that Key 0 allows duplicates, is modifiable, and consists of two segments:

- The first segment starts in position 21, is 20 characters long, allows duplicates, is modifiable, and will be sorted as a STRING type. The dashes indicate that a null value was not defined. The Unique Values column indicates that 16 unique values were inserted for this segment. This segment uses the upper.alt alternate collating sequence file.
The second segment starts in position 7, is 12 characters long, allows duplicates, is modifiable, and will be sorted as a STRING type. Sixteen unique values were inserted for this segment. This segment uses the upper.alt alternate collating sequence file.

Key 1 is the key the database engine uses in logging this file. Key 1 consists of one segment. It starts in position 1, is six characters long, does not allow duplicates, is modifiable, and will be sorted as a STRING type. Sixteen unique values were inserted for this key. This key uses the upper.alt alternate collating sequence file.

Key 2 consists of one segment. It starts in position 83, is 10 characters long, allows duplicates, is modifiable, and will be sorted as a STRING type. Seven unique key values were inserted for this key. This key uses the upper.alt alternate collating sequence file.

**File Version Notes**

When reporting the file format version for a file, the database engine reports the earliest engine version that can read the specified file. For example, you may have a file that was created in Btrieve 5.x format, but it may be reported as a version 3.x file because it does not use any 4.x or 5.x features. Starting with the 6.x format, the file itself contains a version stamp. Prior to 6.x, the only way to determine the file format version of a file is by inspecting the features that it uses. For version 5.x or earlier files, the next table shows the features which, if used, determine the version that is reported for the file:

<table>
<thead>
<tr>
<th>This file format version is reported...</th>
<th>... if one or more of these features are in use:</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.x</td>
<td>Compressed records</td>
</tr>
<tr>
<td></td>
<td>Key only file</td>
</tr>
<tr>
<td>4.x</td>
<td>Extended key types</td>
</tr>
<tr>
<td></td>
<td>Variable length records</td>
</tr>
<tr>
<td></td>
<td>Index added with CreateIndex operation</td>
</tr>
<tr>
<td>3.x</td>
<td>None of the above</td>
</tr>
</tbody>
</table>
Displaying MicroKernel Engine Version

This section includes detailed information about displaying the version of the MicroKernel Engine using the `ver` command.

**Ver**

The `ver` command returns the version number of both the MicroKernel Engine and the Requester (the access module).

**Format**

`butil -ver`

**Remarks**

When you run the `ver` command, the utility displays messages similar to the following:

- The Btrieve Requester version is 11.00.
- The Btrieve Version is 11.00.
Unloading the MicroKernel Engine and Requester (DOS only)

Stop
Use the stop command to unload the MicroKernel Engine and, if applicable, the Requester.

Format
butil -stop
Performing Continuous Operations

The commands pertaining to continuous operations, startbu and endbu, are discussed in the chapter Logging, Backup, and Restore.
Performing Archival Logging

The Maintenance utility (GUI or BUTIL command line) provides a way to roll forward archival log files into the data files. See also the chapter Logging, Backup, and Restore.

The BUTIL **rollfwd** command recovers changes made to a data file between the time of the last backup and a system failure. If a system failure occurs, you can restore the backup copy of your data file and then use the BUTIL **rollfwd** command, which applies all changes stored in the archival log to your restored data files. Do not use this command unless you have restored data files from backup.

**Note** You cannot take advantage of the **rollfwd** command unless you both enable the MicroKernel’s Archival Logging Selected Files option and back up your files before a system failure occurs.

You can also use the **rollfwd** command to produce an output file of logged operations. The **rollfwd** command can produce the output file either before you roll changes forward or at the same time as the roll forward.

You can roll forward a single file, all data files on a volume, all data files on a drive, or a list of files, volumes, and/or drives.

**Using the GUI**

1. Access **Maintenance** from the operating system Start menu or Apps screen or from the Tools menu in PSQL Control Center.
2. Within the Maintenance window, click **Data > Roll Forward** The Roll Forward dialog box appears.

![Roll Forward Dialog](image)

3. Select the specific operation type: single file, list of files, volume name, or drive letter. When you select either volume name or drive letter, you must insert a back slash (\) or forward slash (/) at the end (for example, \server\vol1 or D:\).
4. You can generate a log file, called a dump file, of all the Btrieve operations required to perform the roll forward tasks.
By default, this file is not created. Select the Generate Dump File check box to generate a file. You can also specify the following options.

Table 81 Roll Forward GUI Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only Create Dump File</td>
<td>Indicates that only the dump file is to be created, and the roll forward operation is not to be performed.</td>
</tr>
<tr>
<td>Dump File Name</td>
<td>Contains the name of the dump file, which must begin with a slash and not contain a drive letter or server/volume name.</td>
</tr>
<tr>
<td>Data Buffer Length</td>
<td>Indicates the number of data buffer bytes to write to the dump file for each Btrieve operation.</td>
</tr>
<tr>
<td>Key Buffer Length</td>
<td>Indicates the number of key buffer bytes to write to the dump file for each Btrieve operation.</td>
</tr>
<tr>
<td>Display Numbers as HEX</td>
<td>If you select this option, the numbers in the dump file output are formatted as hexadecimal. If you do not select this check box, the numbers are displayed in decimal format.</td>
</tr>
<tr>
<td>Verbose</td>
<td>Includes additional information like user name, network address, and time stamp in the dump file.</td>
</tr>
</tbody>
</table>

**Note** If the key buffer or the data buffer is not an input parameter for the particular Btrieve operation, nothing is written to the dump file.

5 Click Execute to generate the dump file and/or perform the roll forward operation. If the data is valid, the Roll Forward Status dialog box appears.

Figure 38 Roll Forward Status Dialog Box

As files are processed, they are added to the scrolling list box which displays the file name and the PSQL status code returned from the roll forward operation.

If an error occurs during processing, the Roll Forward Continue on Error dialog box appears. This dialog box allows you to continue without being prompted again, to continue and be prompted again, or to stop processing files.
Using the Command Line

This section explains the syntax for the command line usage of Roll Forward.

```
butil -rollfwd <sourceFile | drive | @listFile> 
  |  
  |  [/E[<keyLength>] /H] /V] [/O<ownerList | owner>|*] 
  |  [/A] [/UID<name> <PWD<word>> [DB<name>]] 
```

**sourceFile**  The fully qualified name of a data file for which to roll forward changes. For Windows platforms, you do not need to specify the name of the path if the data file resides in the same directory as your current directory.

**drive**  A drive letter for which to roll forward changes. End the volume name with a backslash (\) or forward slash (/), as in F:\ or F: /.

**listFile**  The fully qualified name of a text file containing the paths of files, volumes, or drives for which to roll forward changes. Separate these paths with a carriage return/line feed. If the Maintenance utility encounters an error, the utility stops rolling forward the current file, but does not roll back the changes already made. If you specify the /A option, the utility continues rolling forward with the next file.

**/LdumpFile**  Produces an output file, but does not roll forward.

**/WdumpFile**  Rolls forward and produces an output file.

**dumpFile**  The file name of the output file to which the Maintenance utility writes a list of logged operations. The default is \BLOG\BROLL.LST, relative to the root of the physical drive. The file name cannot contain a drive letter or volume name and must start with a forward slash (/) or backslash (\). The Maintenance utility places the file on the same volume as the BLOG.CFG file.

**/TdataLength**  Specifies the length of the operation's data buffer to write to the output file. If you do not specify this option, the utility does not include data buffer contents in the output file.

**/EkeyLength**  Specifies the length of the operation's key buffer to write to the output file. If you do not specify this option, the utility does not include key buffer contents in the output file.

**/H**  Instructs the utility to show numbers in the output file in hexadecimal notation. If you do not specify this option, numbers in the output file are in ASCII format. This option affects the format of the Entry Count, Op Code, Key Number, and Data Length fields.

**/V**  Instructs the utility to include additional information (such as the user name, network address, and time stamp) in the output file.
Performing Archival Logging

Note
If the key buffer or the data buffer is not an input parameter for the particular Btrieve operation, nothing is written to the dump file.

Examples

Example A
The following example recovers changes to the CLASS.MKD file from the default archival log and log location.

`butil -rollfwd file_path\PSQL\Demodata\class.mkd`

(For default locations of PSQL files, see Where are the PSQL files installed? in Getting Started With PSQL.)

Example B
This example recovers changes and outputs them to all files on the d:\ volume with the following options:
- use default dump file
- dump 32 bytes of the data buffer
- dump 4 bytes of the key buffer
- dump in hex mode

`butil -rollfwd d:\ /W /H /T32 /E4`

Example C
The following example does not perform roll forward but only outputs the changes to the files listed in files.txt with the following dump options:
- use d:\temp\files.lst as the dump file
- use verbose mode
Manipulating Btrieve Data Files with Maintenance

- data files have owner names: own123 and own321
- do not dump data or key buffer

butil -rollfwd d:\temp\files.txt /L\temp\files.lst /V /Oown123,own321
Converting Data Files

Maintaining PSQL File Compatibility

PSQL includes tools that convert PSQL files to take advantage of features in the latest versions of the PSQL engines. This chapter describes those tools, why you might need to use them and how to do so. This chapter includes the following sections:

- Rebuild Utility Concepts
- Rebuild Utility GUI Reference
- Rebuild Utility Tasks
Converting Data Files

Rebuild Utility Concepts

The Rebuild utility allows you to perform the following operations on MicroKernel files (data files and dictionary files):

- convert older file formats to a newer PSQL format
- convert newer file formats to a format not older than a 6.x format
- rebuild a file using the same file format (provided the format is 6.x, 7.x, 8.x, or 9.x)
- add file indexes
- change file page size
- rebuild files to include system data and key or system data without key
- specify a location and name of the log file used by Rebuild

If your database uses dictionary files (DDFs), you must rebuild them as well as the data files.

Read further in this section to understand the conceptual aspects of rebuilding data files.

Platforms Supported

File Formats

Command Line Parameters

Temporary Files

Optimizing the Rebuild Process

Log File

For information on using the Rebuild utility, see one of these sections:

- Rebuild Utility GUI Reference
- Rebuild Utility Tasks
- CLI Tasks

Platforms Supported


Linux CLI Rebuild

Rebuild runs as a program, rbldcli, on Linux. By default, the program is located at /usr/local/psql/bin.

Windows CLI Rebuild

Rebuild runs as a program, rbldcli.exe, on Windows. By default, the program is installed in the Program Files directory.
**File Formats**

The current database engines remain compatible with some older data and dictionary file formats, but you may want to convert files to the current format to take advantage of current features. The following table lists the primary reasons for converting from an older to a newer format.

Table 82  Rebuild Utility Conversions

<table>
<thead>
<tr>
<th>Original File Format</th>
<th>Converted File Format</th>
<th>Reason for Conversion</th>
</tr>
</thead>
<tbody>
<tr>
<td>9.0</td>
<td>9.5</td>
<td>More than 119 segment keys and files sizes up to 256 GB.</td>
</tr>
<tr>
<td>8.x</td>
<td>9.x</td>
<td>Add support for file sizes up to 128 GB.</td>
</tr>
<tr>
<td>8.x</td>
<td>8.x</td>
<td>Remove deleted record space from a file, change the page size, or add system data.</td>
</tr>
<tr>
<td>Pre-8.x</td>
<td>8.x</td>
<td>Take advantage of write (insert, update, delete) performance improvements offered by Turbo Write Accelerator.</td>
</tr>
<tr>
<td>7.x</td>
<td>7.x</td>
<td>Original file does not have a system key.</td>
</tr>
<tr>
<td>Pre-7.x</td>
<td>7.x</td>
<td>Take advantage of 7.x features and improve general performance.</td>
</tr>
<tr>
<td>Pre-6.0</td>
<td>6.x</td>
<td>Take advantage of 6.x features and improve general performance. Use this option only if you are still running the 7.x engine with other 6.x engines.</td>
</tr>
</tbody>
</table>

The file format that results from using the command-line Rebuild depends on the `-f` parameter. If you omit the `-f` parameter, Rebuild uses the value set for the MicroKernel's Create File Version configuration option. For example, if the Create File Version value is 8.x, then running the Rebuild utility on version 7.x files converts them to 8.x format. See Create File Version and “-f” parameter.

It is suggested that you back up all the data files you plan to convert before running Rebuild. This is particularly true if you are rebuilding files to the same location as the source files (in which case the rebuilt files replace the source files). Having backup copies allows you to restore the original files if you so desire. To ensure that the backup is successful, you may perform one or more of the following operations:

- Close all data files before running the backup utility.
- Use continuous operations (only during the backup).

⚠️ **Note** You cannot run Rebuild on a file that is in continuous operation mode.

**Temporary Files**

On Windows, Rebuild creates temporary files in the directory specified by the TMP system environment variable. By default on Linux, Rebuild creates temporary files in the output directory (or in the source directory if the `-b` parameter is not used). Therefore, you need enough disk space in the temporary file directory (while the Rebuild utility is running) to potentially accommodate both the original file and the new file. You can specify a different directory for storing these files by using the Output Directory option in the Rebuild GUI version or by using the `-b` parameter with the CLI versions.
Converting Data Files

Normally, Rebuild deletes temporary files when the conversion is complete. However, if a power failure or other serious interruption occurs, Rebuild may not delete the temporary files. If this occurs, delete the following types of temporary files:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Temporary File Names</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux</td>
<td>_rbldxxxxxx, where xxxxxx is six random letters. <strong>Caution:</strong> Ensure that you do not delete the Rebuild executable, rbldcl.</td>
</tr>
<tr>
<td>Windows</td>
<td>_rbldx, where x is a number.</td>
</tr>
</tbody>
</table>

**Optimizing the Rebuild Process**

Rebuild makes Btrieve calls to the database engine. Therefore, the database engine configuration settings and the amount of random access memory (RAM) in your computer affect the performance of the rebuild process. This is particularly evident in the amount of time required to rebuild large data files.

In general, building indexes requires much more time than building data pages. If you have a data file with many indexes, it requires more time to rebuild than would the same file with fewer indexes.

The following items can affect the rebuild processing time:

- **CPU Speed and Disk Speed**
- **Amount of Memory**
- **Sort Buffer Size**
- **Max MicroKernel Memory Usage**
- **Cache Allocation Size**
- **Index Page Size**
- **Number of Indexes**

**CPU Speed and Disk Speed**

The speed of the central processing unit (CPU) and access speed of the physical storage disk can affect processing time during a rebuild. In general, the faster the speed for both of these, the faster the rebuild process. Disk speed is more critical for rebuilding files that are too large to fit entirely in memory.

---

**Tip**

Large files, such as 3 or 4 GB or more, may take several hours to convert. If you have more than one database engine available, you may wish to share the rebuild processing among a number of machine CPUs. For example, you could copy some of your files to each machine that has a database engine installed, then copy the files back after the rebuild process.

---

**Amount of Memory**

Rebuild is capable of rebuilding a file using two different methods, a default method and an alternative method. See `-m<0 | 2>` parameter. The method chosen depends on the amount of memory available. For the default method (`-m2`), Rebuild takes the following steps provided available memory exists.

1. Creates a new, empty data file with the same record structure and indexes as defined in the source file.
2. Drops all the indexes from the new file.
3 Copies all the data into the new file, without indexes.

4 Adds the indexes, using the following process.
   a. For a particular key in the source file, reads as many key values as possible into a memory buffer
      using the Extended Step operation.
   b. Sorts the values in the memory buffer and writes the sorted values to a temporary file.
   c. Repeats steps a and b, processing the key value from every record.
      The temporary file now contains several key value sets, each of which has been individually
      sorted.

5 Merges the sets into index pages, filling each page to capacity. Each index page is added to the data
   file at the end, extending the file length.

6 Repeats steps 4 and 5 for each remaining key.

If any failure occurs during this process, such as a failure to open or write the temporary file, Rebuild
starts over and uses the alternative method to build the file.

Rebuild uses an alternative method (-m0) when insufficient memory exists to use the default method,
or if the default method encounters processing errors.

1 Creates a new, empty data file with the same record structure and indexes as defined in the source
   file.

2 Drops all the indexes from the new file.

3 Copies all the data into the new file, without indexes.

4 Adds the indexes, using the following process.
   a. For a particular key in the source file, reads one record at a time using the Step Next operation.
   b. Extracts the key value from the record and inserts it into the appropriate place in the index. This
      necessitates splitting key pages when they get full.
   c. Repeats steps a and b, processing the key value from every record.

5 Repeats step 4 for each remaining key.

The alternative method is typically much slower than the default method. If you have large data files with
many indexes, the difference between the two methods can amount to many hours or even days. The
only way to ensure that Rebuild uses the default method is to have enough available memory. Several
Configuration settings affect the amount of available memory.

Formulas For Estimating Memory Requirements

The following formulas estimate the optimal and minimum amount of contiguous free memory
required to rebuild file indexes using the fast method. The optimal memory amount is enough memory
to store all merge blocks in RAM. The minimum amount of memory is enough to store one merge block
in RAM.

Key Length = total size of all segments of largest key in the file.
Key Overhead = 8 if key type is not linked duplicate. 12 if key type is linked
duplicate.
Record Count = number of records in the file.

Optimal Memory Bytes = (((Key Length + Key Overhead) * Record Count) + 65536) / 0.6
Minimum Memory Bytes = Optimal Memory Bytes / 30

For example, if your file has 8 million records, and the longest key is 20 bytes (not linked duplicate), the preferred amount of memory is 373.5 M B, or ((( 20 + 8 ) * 8,000,000 ) + 65536 ) / 0.6 = 373,442,560 bytes.

The optimal amount of contiguous free memory is 373.5 M B. If you have at least this much free memory available, the Rebuild process takes place entirely in RAM. Because of the 60% allocation limit, the optimal amount of memory is actually the amount required to be free when the rebuild process starts, not the amount that the rebuild process actually uses. Multiply this optimal amount by 0.6 to determine the maximum amount Rebuild actually uses.

The minimum amount of memory is 1/30th of the optimal amount, 12,448,086 bytes, or 12.45 M B.

The divisor 30 is used because the database engine keeps track of no more than 30 merge blocks at once, but only one merge block is required to be in memory at any time. The divisor 0.6 is used because the engine allocates no more than 60% of available physical memory for rebuild processing.

If you do not have the minimum amount of memory available, Rebuild uses the alternative method to rebuild your data file.

Finally, the memory block allocated must meet two additional criteria: blocks required and allocated block size.

Blocks required must be less than or equal to 30, where:

\[
\text{Blocks Required} = \text{Round Up} \left( \frac{\text{Optimal Memory Bytes}}{\text{Allocated Block}} \right)
\]

Allocated block size must be greater than or equal to:

\[
((2 \times \text{Max Keys} + 1) \times (\text{Key Length} + \text{Key Overhead})) \times \text{Blocks Required}
\]

Assuming a 512-byte page size, and a block of 12.45 M B successfully allocated, the value for blocks required is:

\[
\text{Blocks Required} = \frac{373,500,000}{12,450,000} = 30
\]

The first criteria is met.

The value for allocated block size is:

\[
\text{Max Keys} = \frac{(512-12)}{28} = 18
\]

\[
((2 \times 18 + 1) \times (20 + 8)) \times 9 = 9324
\]

Is Allocated Block (12.5 million bytes) larger than 9324 bytes? Yes, so the second criteria is met. The index keys will be written to a temporary file in 12.45 M B pieces, sorted in memory, and then written to the index.

Sort Buffer Size

This setting specifies the maximum amount of memory that the MicroKernel dynamically allocates and de-allocates for sorting purposes during run-time creation of indexes. See Sort Buffer Size.

If the setting is zero (the default), Rebuild calculates a value for optimal memory bytes and allocates memory based on that value. If the memory allocation succeeds, the size of the block allocated must be at least as large as the value defined for minimum memory bytes. See Formulas For Estimating Memory Requirements.

If the setting is a non-zero value, and the value is smaller than the calculated minimum memory bytes, Rebuild uses the value to allocate memory.
Finally, Rebuild compares the amount of memory that it should allocate with 60% of the amount that is actually available. It then attempts to allocate the smaller of the two. If the memory allocation fails, Rebuild keeps attempting to allocate 80% of the last attempted amount. If the memory allocation fails completely (which means the amount of memory is less than the minimum memory bytes), Rebuild uses the alternative method to rebuild the file.

**Max MicroKernel Memory Usage**

This setting specifies the maximum proportion of total physical memory that the MicroKernel is allowed to consume. L1, L2, and all miscellaneous memory usage by the MicroKernel are included (Relational Engine is not included). See Max MicroKernel Memory Usage.

If you have large files to rebuild, temporarily set Max MicroKernel Memory Usage to a lower percentage than its default setting. Reset it to your preferred percentage after you complete your rebuilding.

**Cache Allocation Size**

This setting specifies the size of the Level 1 cache that the MicroKernel allocates; the MicroKernel uses this cache when accessing any data files. See Cache Allocation Size.

This setting determines how much memory is available to the database engine for accessing data files, not for use when indexes are built.

Increasing Cache Allocation to a high value does not help indexes build faster. In fact, it may slow the process by taking up crucial memory that is now unavailable to Rebuild. When rebuilding large files, decrease the cache value to a low value, such as 20% of your current value but not less than 5 MB. This leaves as much memory as possible available for index rebuilding.

**Index Page Size**

The page size in your file also affects the speed of index building. If Rebuild uses the alternative method, smaller key pages dramatically increase the time required to build indexes. Key page size has a lesser effect on building indexes if Rebuild uses the default method.

Rebuild can optimize page size for application performance or for disk storage.

To optimize for performance (your application accessing its data), Rebuild uses a default page size of 4096 bytes. This results in larger page sizes on physical storage and slower rebuilding times.

For a discussion of optimizing page size for disk storage, see Choosing a Page Size in PSQL Programmer's Guide in the Developer Reference.

Assume that your application has 8 million records, a 20-byte key, and uses a page size of 512 bytes. The MicroKernel places between 8 and 18 key values in each index page. This lessens the amount of physical storage required for each page. However, indexing 8 million records creates a B-tree about seven levels deep, with most of the key pages at the seventh level. Performance will be slower.

If you use a page size of 4096 bytes, the database engine places between 72 and 145 key values in each index page. This B-tree is only about four levels deep and requires many fewer pages to be examined when Rebuild inserts each new key value. Performance is increased but so is the requirement for the amount of physical storage.
**Number of Indexes**

The number of indexes also affects the speed of index building. Generally, the larger the number of indexes, the longer the rebuild process takes. The time required to build the indexes increases exponentially with increasing depth of the B-tree.

**Log File**

Information from a rebuild process is appended to a text log file. By default, the log file is placed in the current working directory.

For the CLI Rebuild, the default file name is rbldcli.log on Windows and Linux. You may specify a location and name for the log file instead of using the defaults. See `-lfile` parameter.

You may examine the log file using a text editor. The information written to the log file includes the following:

- Start time of the rebuild process
- Parameters specified on the command line
- Status code and error description (if an error occurs)
- File being processed
- Information about the processing (such as page size changes)
- Total records processed
- Total indexes rebuilt (if the `-m2` processing method is used)
- End time of the rebuild process
- Status of the process (for example, if the file rebuilt successfully)
Rebuild Utility GUI Reference

This section describes the objects on the Rebuild utility graphical user interface (GUI).

File Options Screen

This screen allows you to add files to the rebuild list.

![Figure 40 Rebuild Utility File Selection]

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selected files</td>
<td>The data and dictionary files listed for rebuilding according to your selections using the Add button.</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td>Add button</td>
<td>Adds a data or dictionary file to the list of files to be rebuilt.</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td>Remove button</td>
<td>Removes the selected data or dictionary file in the list.</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td>Clear button</td>
<td>Clears the entire list of selected data and dictionary files.</td>
<td>To rebuild a file or files</td>
</tr>
</tbody>
</table>

Rebuild Options Screen

This screen allows you to select the options for rebuilding files.

![Figure 41 Rebuild Utility File Options]
<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>System Data</td>
<td>Specifies whether you want Rebuild to create a System Data key in the file. System data is necessary</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td></td>
<td>for transaction durability logging. Specifies whether the file is rebuilt with System Data or System</td>
<td></td>
</tr>
<tr>
<td></td>
<td>and Key Data. The MicroKernel cannot perform logging for a file without system data when no</td>
<td></td>
</tr>
<tr>
<td></td>
<td>user-defined unique key exists.</td>
<td></td>
</tr>
<tr>
<td>Page Compression</td>
<td>Specifies if you want page compression for the file. The choices are “on” (yes), “off” (no), and</td>
<td>Page compression requires a file format of 9.5 or newer.</td>
</tr>
<tr>
<td></td>
<td>“keep existing.” Keep existing retains whatever page compression the file contains, if any.</td>
<td>Record and Page Compression</td>
</tr>
<tr>
<td>Record</td>
<td>Specifies if you want record compression for the file. The choices are “on” (yes), “off” (no), and</td>
<td>Record and Page Compression.</td>
</tr>
<tr>
<td>Compression</td>
<td>“keep existing.” Keep existing retains whatever record compression the file contains, if any.</td>
<td></td>
</tr>
<tr>
<td>Continue on Error</td>
<td>Determines whether the Rebuild utility continues if it encounters an error during the rebuild process.</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td></td>
<td>If you select Yes, the utility continues with the next file even if an error occurs. The utility</td>
<td></td>
</tr>
<tr>
<td></td>
<td>notifies you of non-MicroKernel data files or other errors but continues rebuilding data files. If</td>
<td></td>
</tr>
<tr>
<td></td>
<td>you select No, the utility halts the rebuild if it encounters an error.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This option is useful if you have specified wildcard characters for the rebuilt files.</td>
<td></td>
</tr>
<tr>
<td>Save Settings on Exit</td>
<td>Saves the current values in this dialog box for use in subsequent Rebuild sessions.</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td>Key Number</td>
<td>Specifies the key by which the utility reads when rebuilding a file. If you specify NONE for this</td>
<td>• To rebuild a file or files</td>
</tr>
<tr>
<td></td>
<td>option, the utility clones the files, drops the indexes, copies the records into the new files, and</td>
<td>• Key Attributes in PSQL Programmer's Guide.</td>
</tr>
<tr>
<td></td>
<td>rebuilds the indexes. Because this method is faster and creates smaller files than specifying a key</td>
<td></td>
</tr>
<tr>
<td></td>
<td>number, use it whenever possible.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This method may create a new file in which the records are in a different physical order than in the</td>
<td></td>
</tr>
<tr>
<td></td>
<td>original file.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you specify a key number, the utility clones and copies the files without dropping and replacing</td>
<td></td>
</tr>
<tr>
<td></td>
<td>indexes. While this method is slower than specifying NONE, it is available in case you do not want</td>
<td></td>
</tr>
<tr>
<td></td>
<td>to rebuild your indexes.</td>
<td></td>
</tr>
<tr>
<td>File Format</td>
<td>Previous, the Rebuild utility built the file version based on the Create File Version configuration</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td></td>
<td>setting.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>This Rebuild utility allows you to specify the file version type independent of that setting.</td>
<td></td>
</tr>
<tr>
<td>Page Size</td>
<td>Specifies the page size (in bytes) of the new files. Choose either EXISTING, Optimal (disk space),</td>
<td>• To rebuild a file or files</td>
</tr>
<tr>
<td></td>
<td>Optimal (data access), or a size in bytes. If you select EXISTING, the utility uses the existing page</td>
<td>• For optimizing for data access, see Optimizing the Rebuild Process</td>
</tr>
<tr>
<td></td>
<td>size. The utility changes the page size if the original size does not work. For example, assume you</td>
<td>• For optimizing for disk space, see Choosing a Page Size in PSQL Programmer's Guide.</td>
</tr>
<tr>
<td></td>
<td>have a v5.x file with a page size of 1,024 and 24 keys. Because Btrieve 6.0 and later supports only 23</td>
<td></td>
</tr>
<tr>
<td></td>
<td>keys for a page size of 1,024, the utility automatically selects a new page size for the file and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>writes an informative message to the status file.</td>
<td></td>
</tr>
</tbody>
</table>
Rebuild Progress Screen

This screen allows you to see the rebuild progress and to view the log file after the rebuild process completes.

![Rebuild Utility Progress Screen](image)

<table>
<thead>
<tr>
<th>GUI Object</th>
<th>Description</th>
<th>Related Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Output Path</td>
<td>Specifies an alternate location for the rebuilt files. (The default location is the current directory.) You must specify a directory that already exists. This option lets you rebuild large files on a different server. The MicroKernel and its communications components must be loaded on the server that contains the rebuilt files. Do not use wildcard characters in the path. If the Output Directory location is different than the original file’s location, the original file is not deleted during the rebuild. If the output directory is the same as the original file, the original file is deleted upon completion of the rebuild. DefaultDB w/ DB security: Cannot rebuild outside DB’s file locations in Maintain Named Databases</td>
<td>To rebuild a file or files</td>
</tr>
<tr>
<td>Log File</td>
<td>Specifies a location for the rebuild log file. (The default location is the current working directory.) Do not use wildcard characters in the path.</td>
<td>• To rebuild a file or files • Log File</td>
</tr>
</tbody>
</table>

**GUI Object**

- **Message area**
  - Displays the information about the file being rebuilt, or a summary of the operations if rebuilding has completed.
- **View Log File**
  - Allows you to see information about the rebuild process for each file. Click **View Log File** to display the rebuild log using your default text viewer.
Rebuild Utility Tasks

The following Rebuild tasks are available:

- GUI Rebuild: GUI Tasks
- Command-line Rebuild: CLI Tasks

GUI Tasks

- To start the GUI Rebuild utility
- To rebuild a file or files

➢ To start the GUI Rebuild utility

Click Tools then Rebuild from the PSQL Control Center menu or access Rebuild from the operating system Start menu or Apps screen.

➢ To rebuild a file or files

1. After you click Next at the Rebuild welcome screen, the Select Files screen appears.
2. Click Add and select the data or dictionary file you want to rebuild. You can select more than one file to rebuild at a time.

![Select Files Dialog Box](image)

The Rebuild utility deletes the original file after rebuilding it if the file is being rebuilt in the same directory. If the new file is in a different directory, the original file is not deleted.

3. Click Next after you have added the desired file or files.
4. Specify the rebuild options. See Rebuild Options Screen.
5. Click Next to begin the rebuild process.

The utility reports the processing information. When the rebuild process completes, the success or failure of it displays and View Log File is enabled.
6 To display the results, click **View Log File**. The contents of the log file display in the default text editor for the operating system.

The Rebuild utility writes to the log file for every file it attempts to convert. If you disabled the **Continue on Error** setting, the log file contains the information up to the point of the error. If the rebuild was not successful, the status file contains error messages explaining why the rebuild failed.

7 Click **Finish** when you have finished rebuilding files and viewing the log file.

**CLI Tasks**

The Rebuild command-line utility is named rbldcli.exe on Windows and rbldcli on Linux. The following command line Rebuild utility tasks are available:

- To run Rebuild on Linux
- To run Rebuild on Windows
- To see your progress while rebuilding files
- To learn about the command line parameters for the CLI Rebuild utility, see the following section.

**Command Line Parameters**

The parameter option specifies the parameter(s) used with the utility. You may use the parameters in any order. Precede each parameter with a hyphen (-). Do not place a space after the hyphen or after the single-letter parameter and the parameter value.

**Note** On Linux platforms only, the parameters are case sensitive.
Converting Data Files

Parameter is defined as follows:

- **-c**
  Instructs Rebuild to continue with the next data or dictionary file if an error occurs. The utility notifies you of non-MicroKernel data files or errors with MicroKernel files, but continues rebuilding data files. The errors are written to the log file. See Log File.

  **Tip:** This parameter is particularly useful if you specify wildcard characters ("*") for a mixed set of files. Mixed set means a combination of MicroKernel files and non-MicroKernel files. Rebuild reports an error for each non-MicroKernel file (or any errors on MicroKernel files), but continues processing.

- **-d**
  If you specify -d, Rebuild converts pre-6.0 supplemental indexes (which allow duplicates) to 6.x, 7.x, or 8.x indexes with linked-duplicatable keys.

  If you omit this parameter, Rebuild preserves the indexes as repeating-duplicatable keys.

  If you access your data files only through the MicroKernel Engine and your files have a relatively large number of duplicate keys, you can use the -d parameter to enhance the performance of the Get Next and Get Previous operations.

- **-m<0 | 2>**
  The "m" parameter stands for "method." Rebuild selects a processing method whether you specify this parameter or not. If you omit this parameter, Rebuild does the following:
  
  • uses -m2 as the default method if sufficient available memory exists
  • uses an alternative method,-m0, if the amount of available memory is not sufficient.

  See Amount of Memory for how the amount of memory affects the method chosen.

  0
  Clones and copies the data or dictionary file without dropping and replacing indexes. This method is slower than the -m2 method. It is available in case you do not want to rebuild your indexes.

  A file built with the -m0 creates a file where each key page is about 55% to 65% full. The file is more optimized for writing and less for reading. If you can afford the extra rebuild time, which can be considerable depending on the situation, you might want to rebuild a file optimized for writing.

  See also Optimizing the Rebuild Process.

  2
  Clones the data or dictionary file, drops the indexes, copies the records into the new file, and rebuilds the indexes. This method is faster and creates smaller files than the -m0 method.

  The -m2 method may create a new file in which the records are in a different physical order than in the original file.

  A file built with the -m2 method has key pages that are 100% full. This allows the file to be optimized for reading.

- **-p<D | P | bytes>**
  • Optimizes page size for disk storage or processing, or specifies a specific page size to use for the rebuilt file.

  If you omit this parameter, Rebuild uses the page size from the source file. If the source page size does not work for the current database engine, Rebuild changes the page size and displays an informative message explaining the change. (For example, older file formats, such as 5.x, supported a page size of 1024 with 24 keys. File format 8.x supports only 23 keys for a page size of 1024, so Rebuild would select a different page size if building an 8.x file.)

  The database engine may ignore the page size specified and automatically upgrade the page size. For example, for the 9.5 file format, the odd page sizes such as 1536 and 3072 are not supported. The database engine automatically upgrades to the next valid page size because the next valid page size is more efficient. For older file formats, the database engine may upgrade the page size based on other conditions.

  See also Index Page Size.
Rebuild Utility Tasks

D Optimizes page size for disk storage.


P Optimizes for processing (that is, for your application accessing its data). For -pP, Rebuild uses a default page size of 4096 bytes.

See Optimizing the Rebuild Process

(bytes) Specifies the page size (in bytes) for the new file. For file versions prior to 9.0, the valid values are 512, 1024, 1536, 2048, 2560, 3072, 3584, and 4096. For file version 9.0, the values are the same with the addition of 8192. For file version 9.5 or newer, the valid values are 1024, 2048, 4096, 8192, and 16384.

-b dirname Specifies an alternate location for the rebuilt file (which may also be a location on a different server). The default location is the directory where the data file is located. You must specify a location that already exists. Rebuild does not create a directory for you. The directory also must be on a machine that is running the PSQL database engine.

You may use either a fully qualified path or a relative path. Do not use wildcard characters in dirname.

On your local server, the MicroKernel Database Engine and the Message Router must be loaded. On a remote server, the MicroKernel Database Engine and communications components must be loaded.

If you omit this parameter, the rebuilt file replaces the original data file. A copy of the original file is not retained.

If you specify this parameter, the rebuilt file is placed in the specified location and the original file is retained. An exception to this is if the specified location already contains data files with the same names. Rebuild fails if the alternate location you specify contains files with the same names as the source files. For example, suppose you want to rebuild mydata.mkd, which is in a directory named folder1. You want to place the rebuilt file into a directory named folder2. If mydata.mkd also exists in folder2 (perhaps unknown to you), Rebuild fails and informs you to check the log file.

Note: Ensure that you have create file permission for the location you specify (or for the location of the source file if you omit the parameter).

-k number Specifies the key number that Rebuild reads from the source file and uses to sort the rebuilt file. If you omit this parameter, Rebuild reads the source file in physical order and creates the rebuilt file in physical order.

See also Optimizing the Rebuild Process.

-s[D | K] Retains in the rebuilt file the existing system data and key from the source file. If you omit this parameter, Rebuild does not include the system data and key in the rebuilt file.

See also System Data.

D Rebuilds the file to include system data. The system data is not indexed. See also System Data.

K Rebuilds the file to include system data and key. The system data is indexed. See also System Data.
Converting Data Files

-`-f<6 | 7 | 8 | 9 | 95>` Specifies a file format for the rebuilt data or dictionary file. File formats supported are versions 6.x, 7.x, 8.x, and 9.x. The following example rebuilds a file to the 9.0 format:

```
rbldcli -f9 file_path\class.mkd
```

The following example rebuilds a file to the 9.5 format:

```
rbldcli -f95 file_path\class.mkd
```

If you omit this parameter, Rebuild uses the value set for the MicroKernel's “Create File Version” configuration option. See `Create File Version`.

**Note1:** If you specify a file format newer than the version supported by the current database engine, Rebuild uses the highest supported file format of that engine. Rebuild reports no error or message for this.

**Note2:** Rebuild does not convert data types in indexes. If you rebuild a file to an older file format for use with an older database engine, ensure that the engine supports the data types used. You must manually adjust data types as required by your application and by the database engine.

Example 1. Your data file contains index fields that use the WZSTRING data type. If you rebuild the data file to a 6.x file format, the WZSTRING data type is not converted. You would be unable to use the data file with a Btrieve 6.15 engine. That engine does not support the WZSTRING data type.

Example 2. Your data file contains true NULLs. You rebuild the data file to a 7.x file format. The true NULLs are not converted. You would be unable to use the data file with the PSQL 7 engine. That engine does not support true NULLs.

-`-uuname` Specifies the name of the user authorized to access a database with security enabled.

-`-pwdpword` Specifies the password for the user who is identified by `uname`. `Pword` must be supplied if `uname` is specified.

-`-dbdbname` Specifies the name of the database on which security is enabled.
Rebuild Utility Tasks

File and @command_file are defined as follows:

**file**
Specifies the data and dictionary file(s) to convert. If the source file is not in the current working directory, include the location, either as a fully qualified path or as a relative path. You may use the asterisk (*) wildcard character in the file name to specify multiple files.

**Note**: If the original file contains an owner name, Rebuild applies the owner name and level to the rebuilt file.

**@command_file**
Specifies a command file for Rebuild to execute. You may include multiple entries in one command file. Each entry in the command file contains the command line parameters (if any) and the set of files to convert, followed by <end> or [end].

When specifying the files to convert, use full directory names. You may use the asterisk (*) wildcard character in the file names.

The following is an example of a Rebuild command file:

```
-c d:\mydir\*. * <end>
-c -p1024 e:\dir\*. * <end>
-m0 -k0 d:\ssql\*. * <end>
```

➢ To run Rebuild on Linux

1. Ensure that the account under which you are logged in has permission to run PSQL utilities.

   By default, you must be logged in as user `psql` to run utilities. User `psql` has no password and can be accessed only through the `root` account by using the `su` command. To use utilities from accounts other than `psql`, you must first make modifications to your `.bash_profile`. See PSQL Account Management on Linux in Getting Started With PSQL.

2. Change directory to `/usr/local/psql/bin` directory.

3. Type one of the following commands at the prompt:

   ```
   rbldcli [-parameter ...] file
   or
   rbldcli @command_file
   ```

Parameter, file, and @command_file are defined in Command Line Parameters.

**Example Usage**

The following example continues on error, sets a page size of 4096 bytes, and places the rebuilt files in a different directory on the server.

```
rbldcli -c -p4096 -b /usr/local/psql/tmp /usr/local/psql/data/DEMODATA/* .mkd
```

➢ To run Rebuild on Windows

1. Open a command prompt on the machine where PSQL is installed.

2. Optionally, change to the `\bin` directory where you installed the Program Files. (This is not required if the location is in the Path system variable.)

3. Type one of the following commands at the prompt:

   ```
   rbldcli [-parameter ...] file
   or
   rbldcli @command_file
   ```
Converting Data Files

Parameter, file, and @command_file are defined in Command Line Parameters.

Example Usage
The following example continues on error, sets a page size of 4096 bytes, and places the rebuilt files in a different directory on the server.

    rbldcli -c -p4096 -bc:\dbtemp c:\datafiles\*.mkd

➢ To see your progress while rebuilding files
Rebuild reports on the screen the number of records processed per file, incrementing 50 records at a time. In addition, Rebuild writes information to a text log file. See Log File.
Description Files

Using Description Files to Store Btrieve File Information

A description file is an ASCII text file that contains descriptions of file and key specifications that the Maintenance utility can use to create data files and indexes. Some users employ description files as a vehicle for archiving information about the data files they have created. Description files are not the same as DDFs, or Data Dictionary Files, used by the Relational Engine.

Description files contain one or more elements. An element consists of a keyword, followed by an equal sign (=), followed by a value (with no space). Each element in a description file corresponds to a particular characteristic of a data file or key specification.

Note Before using description files, you should be familiar with Btrieve fundamentals. For information about these topics, refer to the PSQL Programmer’s Guide.

This appendix discusses the following topics:

- Rules for Description Files
- Description File Examples
- Description File Elements
Rules for Description Files

Use the following rules when creating a description file.

- Enter elements in either uppercase or lowercase.
- Separate elements from each other with a separator (blank space, tab, or carriage return/line feed), as in the following example:
  
  ```
  record=4000
  key=24
  ```
- Specify the description file elements in the proper order. Table 83 presents the elements in the appropriate order.
- Address all element dependencies. For example, if you specify `nullkey=allsegs` in your description file, you must also specify a value for the `value=` element.
- Define as many keys as you specify with the Key Count element. For example, if you specify `key=12`, you must define 12 keys in the description file.
- For a key that consists of multiple segments, you must define the following elements for each key segment:
  - Key Position
  - Key Length
  - Duplicate Key Values
  - Modifiable Key Values
  - Key Type

  The Descending Sort Order element is optional for each segment.

- If any key in the file uses an ACS, you must specify an ACS file name, a country ID and code page ID, or an ISR table name. You can include this information as either the last element of the key (applies to current key only) or the last element in the description file (applies to entire data file).
  - You can specify only one ACS per key, and you must provide an ACS file name, country ID and code page ID, or an ISR table name. Different keys in the same file can use different types of ACSs; for example, Key 0 can use an ACS file name, and Key 1 can use a country ID and code page ID.
  - Different segments of the same key cannot have different ACSs.
  - If you specify an ACS at the end of a description file, it is used as the default ACS. That is, if you specify `alternate=y` for a given key but do not include an ACS file name, country ID and code page ID, or an ISR table name for that key, the database engine uses the ACS file name, country ID and code page ID, or ISR table name specified at the end of the file.
  - If you are creating a new key and you specify `alternate=y` but you omit the ACS file name, country ID and code page ID, or ISR table name, the database engine does not create the key.
- If a Description File element is optional, you can omit it.
- Make sure the description file contains no text formatting characters. Some word processors embed formatting characters in a text file.
Description File Examples

The sample description files shown in this section describe a data file. This data file has a page size of 512 bytes and 2 keys. The fixed-length portion of the record is 98 bytes long. The file allows variable-length records but does not use blank truncation.

The file uses record compression, allows for Variable-tail Allocation Tables (VATs), and has the free space threshold set to 20 percent. The MicroKernel Database Engine preallocates 100 pages, or 51,200 bytes, when it creates the file. The file has two keys: Key 0 and Key 1. Key 0 is a segmented key with two segments.

In Figure 45, both keys use the same ACS file name (upper.alt). In Figure 46, both keys use the same country ID (-1) and code page ID (-1). In Figure 47, Key 0 and Key 1 use different ACS file names (lower.alt and upper.alt, respectively). In Figure 48, the file has no keys except the system-defined key used for logging.

Figure 45  Sample Description File Using Alternate Collating Sequence File Name

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y

position=1 length=5 duplicates=y modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=y

position=6 length=10 duplicates=y modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=n

position=16 length=2 duplicates=n modifiable=y type=numeric descending=y
nullkey=n segment=n

name=c:\myacsfiles\upper.alt
```

Figure 46  Sample Description File Using Alternate Collating Sequence ID

```
record=98 variable=y truncate=n compress=y
key=2 page=512 allocation=100 replace=n
fthreshold=20 vats=y

position=1 length=5 duplicates=y modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=y

position=6 length=10 duplicates=y modifiable=n type=string alternate=y
nullkey=allsegs value=20 segment=n

position=16 length=2 duplicates=n modifiable=y type=numeric descending=y
nullkey=n segment=n

countryid=-1 codepageid=-1
```
Figure 47  Sample Description File Using Alternate Collating Sequence File Name on a Key Segment

```plaintext
record=98 variable=y truncate=n compress=y key=2 page=512 allocation=100 replace=n fthreshold=20 vats=y
position=1 length=5 duplicates=y modifiable=n type=string alternate=y nullkey=allsegs value=20 segment=y
name=c:\myacsfiles\upper.alt

position=6 length=10 duplicates=y modifiable=n type=string alternate=y nullkey=allsegs value=20 segment=n
name=c:\myacsfiles\lower.alt

position=16 length=2 duplicates=n modifiable=y type=numeric descending=y nullkey=n segment=n
name=c:\myacsfiles\upper.alt
```

Figure 48  Sample Description File Using System-Defined Key for Logging

```plaintext
record=98 variable=y truncate=n compress=y key=2 page=512 allocation=100 replace=n fthreshold=20 vats=y sysdataonrecord=loggable
```
Description File Elements

Description file elements must appear in a particular order. Table 83 lists the description file elements in the appropriate order. For each element, the table specifies the required format and the range of acceptable values.

- An asterisk (*) indicates that the element is optional.
- A pound sign (#) indicates that it is not applicable in the current MicroKernel version but is retained for backward compatibility with previous MicroKernel versions.
- A percent sign (%) indicates that the element is applicable only to the current MicroKernel version.

Table 83 Summary of Description File Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Keyword and Format</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>File Specification</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Comment Block*</td>
<td>/* . . . . . . . . */</td>
<td>5,120 bytes</td>
<td>None.</td>
</tr>
<tr>
<td>Record Length</td>
<td>record=nnnn</td>
<td>4 – 8,184</td>
<td>None.</td>
</tr>
<tr>
<td>Variable-Length Records</td>
<td>variable=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Reserved Duplicate Pointer*</td>
<td>dupkey=&lt;nnn&gt;</td>
<td>0 – 119</td>
<td>Applicable only to files for which you plan to add linked-duplicatable keys.</td>
</tr>
<tr>
<td>Blank Truncation*</td>
<td>truncate=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Record Compression*</td>
<td>compress=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Key Count</td>
<td>key=nnn</td>
<td>0 – 119</td>
<td>Specify 0 to create a data-only file. If key count is 0, then Include Data and Use System Data cannot be set to &quot;no.&quot;</td>
</tr>
<tr>
<td>Page Size</td>
<td>page=nnnn</td>
<td>512 – 4096 bytes for file versions prior to 9.0 (a multiple of 512 bytes up to 4096) 512, 1024, 1536, 2048, 2560, 4096; or 8192 bytes for file version 9.0. 1024, 2048, 4096, 8192, or 16384 bytes for file versions 9.5 and newer.</td>
<td></td>
</tr>
<tr>
<td>Page Preallocation*</td>
<td>allocation=nnnnn</td>
<td>1 – 65,535</td>
<td>None.</td>
</tr>
<tr>
<td>Replace Existing File*#</td>
<td>replace=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Include Data*</td>
<td>data=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
</tbody>
</table>
Table 83  Summary of Description File Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Keyword and Format</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free Space Threshold*</td>
<td>fthreshold=&lt;5</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>Variable-Tail Allocation Tables (VATs)</td>
<td>huge=&lt;y</td>
<td>n&gt; # vats=&lt;y</td>
<td>n&gt;</td>
</tr>
<tr>
<td>Balanced Index*</td>
<td>balance=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Use Key Number *</td>
<td>usekeynum=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>^Use System Data*%</td>
<td>sysdataonrecord= &lt;n</td>
<td>loggable&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Page Compression*</td>
<td>pagecompress=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Key Specification Information</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key Number *</td>
<td>keynum=nnn</td>
<td>0 – 118</td>
<td>Must be unique to the file, specified in ascending order, and valid for the file’s Page Size. Applicable only when creating a file.</td>
</tr>
<tr>
<td>Key Position</td>
<td>position=nnnn</td>
<td>1 – 8,184</td>
<td>Cannot exceed the Record Length.</td>
</tr>
<tr>
<td>Key Length</td>
<td>length=nnn</td>
<td>key type limit</td>
<td>Cannot exceed the limit dictated by the Key Type. For binary keys, the key length must be an even number. The total of the Key Position and Key Length cannot exceed the file’s Record Length.</td>
</tr>
<tr>
<td>Duplicate Key Values</td>
<td>duplicates=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Modifiable Key Values</td>
<td>modifiable=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Key Type</td>
<td>type=validMKDEKeyType</td>
<td>N/A</td>
<td>Can enter the entire word (as in float) or just the first three letters (as in flo).</td>
</tr>
<tr>
<td>Descending Sort Order*</td>
<td>descending=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Alternate Collating Sequence</td>
<td>alternate=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
</tbody>
</table>
### Table 83  Summary of Description File Elements

<table>
<thead>
<tr>
<th>Element</th>
<th>Keyword and Format</th>
<th>Range</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case-Insensitive Key*</td>
<td>caseinsensitive=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Repeating Duplicates*</td>
<td>repeatdup=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Null Segments*</td>
<td>nullkey=&lt;allsegs</td>
<td>n</td>
<td>anyseg</td>
</tr>
<tr>
<td>Null Key Value</td>
<td>value=nn</td>
<td>1-byte hex</td>
<td>Used with the Null Segments element.</td>
</tr>
<tr>
<td>Segmented Key</td>
<td>segment=&lt;y</td>
<td>n&gt;</td>
<td>N/A</td>
</tr>
<tr>
<td>Alternate Collating Sequence File Name/ID</td>
<td>name=sequenceFile or countryid=nnn and codepageid=nnn isr=table name (%)</td>
<td>valid path or values valid to operating system or -1</td>
<td>Used with the Alternate Collating Sequence element.</td>
</tr>
</tbody>
</table>

1When the database engine adds a system key, the resulting records may be too large to fit in the file's existing page size. In such cases, the database engine automatically increases the file's page size to the next accommodating size.
Description Files