

Pervasive PSQL v9

What's New in Pervasive PSQL

An Overview of New Features and Changed Behavior

Pervasive Software Inc.
12365 Riata Trace Parkway
Building B
Austin, TX 78727 USA

Telephone: 512 231 6000 or 800 287 4383

Fax: 512 231 6010

Email: info@pervasive.com

Web: <http://www.pervasive.com>



disclaimer

PERVASIVE SOFTWARE INC. 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. PERVASIVE SOFTWARE INC. MAKES NO OTHER WARRANTIES WHATSOEVER, EITHER EXPRESS OR IMPLIED, REGARDING THE SOFTWARE OR THE CONTENT OF THE DOCUMENTATION; PERVASIVE SOFTWARE INC. HEREBY EXPRESSLY STATES AND YOU OR YOUR COMPANY ACKNOWLEDGES THAT PERVASIVE SOFTWARE INC. 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, Pervasive, Pervasive Software, and the Pervasive Software logo are registered trademarks of Pervasive Software Inc.

Built on Pervasive Software, DataExchange, MicroKernel Database Engine, MicroKernel Database Architecture, Pervasive.SQL, Pervasive PSQL, Solution Network, Ultralight, and ZDBA are trademarks of Pervasive Software Inc.

Microsoft, MS-DOS, Windows, Windows 95, Windows 98, Windows NT, Windows Millennium, Windows 2000, Windows XP, Win32, Win32s, and Visual Basic are registered trademarks of Microsoft 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.

Sun, Sun Microsystems, Java, all trademarks and logos that contain Sun, Solaris, or Java, are trademarks or registered trademarks of Sun Microsystems.

All other company and product names are the trademarks or registered trademarks of their respective companies.

© Copyright 2006 Pervasive Software Inc. 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 Pervasive Software. Pervasive Software 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.

What's New in Pervasive PSQL

March 2006

100-004297-001

Contents

About This Manual	vii
Who Should Read This Manual	viii
Manual Organization	ix
Conventions	x
1 What's New in Pervasive PSQL v9	1-1
<i>An Overview of New Features</i>	
List of New Features and Improvements	1-2
COBOL Support	1-3
SQL Enhanced Access	1-3
Shifted and Unshifted Sign Value for NUMERIC Data Types	1-3
Configuration Changes	1-4
I*net Data Server Removed	1-4
5.x Files Support Reduced	1-9
Configuration Settings Not Manually Adjustable	1-10
Configuration Settings With Modified Defaults	1-10
Configuration Settings With Modified Values	1-11
Configuration Settings Not Requiring an Engine Restart	1-11
Configuration Settings That Dynamically Increase	1-11
Windows 16-bit Client Configuration	1-11
File Size Support	1-13
Maximum File Size of 128 Gigabytes	1-13
File Segmentation	1-13
Automatic Upgrade of File Version	1-14
Installation	1-15
Improved Installation Modularity	1-15
Linux	1-16
Performance Equality with Windows	1-16
Client Use of idshosts File and PIDS Protocol Strings	1-16
Network Communications	1-17
Programming Interfaces	1-18
Btrieve API Improvements	1-18
Distributed Tuning Interface Improvements	1-21

Relational Interface Support	1-31
ALTER (rename)	1-31
ALTER TABLE Enhancements	1-31
Bitwise Operators	1-32
CASE Expression	1-32
DEFAULT Keyword.	1-32
EXECUTE Statement	1-33
FROM Clause in UPDATE and DELETE Statements	1-33
Global Variables	1-33
Grouped View Support	1-34
Linked Duplicate Keys for CREATE TABLE	1-34
ODBC 3.5 Support	1-34
Promotion of Numeric Data Types	1-34
Scalar Functions	1-35
SELECT Statement	1-36
SET ANSI_PADDING	1-36
Subquery Expressions.	1-36
System Stored Procedures	1-37
User Defined Functions.	1-37
Utilities and Documentation	1-38
Pervasive PSQL Control Center	1-38
Command Line Interface Utilities	1-39
JavaHelp Documentation.	1-39

Tables

1-1	Reconfiguration Required to Transition from an IDS Environment	1-5
1-2	Configuration Settings Not Manually Adjustable	1-10
1-3	Configuration Settings With Modified Defaults.	1-10
1-4	Configuration Settings With Modified Values.	1-11
1-5	Configuration Settings Not Requiring an Engine Restart.	1-11
1-6	Create Operation Subfunctions	1-19
1-7	Create Subfunctions - use of URI Parameters in Key Buffer	1-20
1-8	Create Subfunctions - use of URI Parameters in Data Buffer	1-20
1-9	New Error Messages for PvConnectServer()	1-30

About This Manual

This manual contains information about the features and enhancements that are new in this release of Pervasive PSQL. This release is referred to as Pervasive PSQL v9. The internal version number is 9.00.

This manual describes the new and changed behaviors of the product relative to the most recent previous release, which is Pervasive.SQL V8 SP2 (8.60).

Who Should Read This Manual

This document is designed for any user who is familiar with Pervasive PSQL and wants to know what has changed in this release of the software.

This manual does not provide comprehensive usage instructions for the software. Its purpose is to explain what is new and different in this particular release of the product.

Pervasive Software Inc. 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 <http://www.pervasive.com/devtalk>.

Manual Organization

This manual begins with an overview of the new features, then provides links to chapters containing additional details where appropriate. *What's New in Pervasive PSQL* is divided into the following sections:

- Chapter 1—“What's New in Pervasive PSQL v9”

This chapter provides an overview of the changes in this release of the software.

This manual also contains an index.

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, etc.
Monospaced font	Monospaced font is reserved for words you enter, such as command syntax.
[]	Square brackets enclose optional information, as in [<i>log_name</i>]. If information is not enclosed in square brackets, it is required.
	A vertical bar indicates a choice of information to enter, as in [<i>file_name</i> @ <i>file_name</i>].
< >	Angle brackets enclose multiple choices for a required item, as in /D=<5 6 7>.
<i>variable</i>	Words appearing in italics are variables that you must replace with appropriate values, as in <i>file_name</i> .
...	An ellipsis following information indicates you can repeat the information more than one time, as in [<i>parameter</i> ...].
::=	The symbol ::= means one item is defined in terms of another. For example, a::=b means the item <i>a</i> is defined in terms of <i>b</i> .

What's New in Pervasive PSQL v9

1

An Overview of New Features

The purpose of this chapter is to summarize and explain the major new features and differences in behavior between this product and the previous release. If further information is available for a given change or feature, a link to that information is provided.

List of New Features and Improvements

This release offers new features and improvements over the last release of the product in the following areas:

- “List of New Features and Improvements” on page 1-2
- “COBOL Support” on page 1-3
- “Configuration Changes” on page 1-4
- “File Size Support” on page 1-13
- “Installation” on page 1-15
- “Linux” on page 1-16
- “Network Communications” on page 1-17
- “Programming Interfaces” on page 1-18
- “Relational Interface Support” on page 1-31
- “Utilities and Documentation” on page 1-38

These features are described in the sections that follow. Also see the README file for additional information about this release that may not be contained in this document.

COBOL Support

Pervasive PSQL v9 provides the following support for COBOL applications:

- “SQL Enhanced Access” on page 1-3
- “Shifted and Unshifted Sign Value for NUMERIC Data Types” on page 1-3

SQL Enhanced Access

The Pervasive PSQL relational interface has been enhanced to include support for COBOL OCCURS constructs, partial REDEFINES, and variable record layouts. No changes are required to your COBOL application to take advantage of the SQL enhanced access.

You enable SQL access by describing the application’s understanding of data to the Pervasive PSQL relational interface. In developer’s terms, you define the metadata to the relational interface.

Refer to the chapter “SQL Access for COBOL Applications” on page D-1 in *SQL Engine Reference* for a complete discussion of how to take advantage of the SQL enhanced access.

Shifted and Unshifted Sign Value for NUMERIC Data Types

Each digit of a NUMERIC data type occupies one byte. The rightmost byte of the number includes an embedded sign with an EBCDIC value. By default, the sign value for positive NUMERIC data types is an unsigned numeric number.

You may now specify that you want to shift the value of the sign for positive NUMERIC data types.

See “NUMERIC” on page A-30 in *SQL Engine Reference* for a complete discussion of this feature.

Configuration Changes

Changes made to configuration for Pervasive PSQL v9 include the following:

- “I*net Data Server Removed” on page 1-4
- “5.x Files Support Reduced” on page 1-9
- “Configuration Settings Not Manually Adjustable” on page 1-10
- “Configuration Settings With Modified Defaults” on page 1-10
- “Configuration Settings With Modified Values” on page 1-11
- “Configuration Settings Not Requiring an Engine Restart” on page 1-11
- “Configuration Settings That Dynamically Increase” on page 1-11
- “Windows 16-bit Client Configuration” on page 1-11

I*net Data Server Removed

I*net Data Server (IDS) is no longer a separately installable component of Pervasive PSQL. Most of the features that were part of IDS are now integrated into the main product. With the removal of the IDS components, you have the following considerations:

- Environment Configuration
- Application Changes

Environment Configuration

If your client/server environment previously used IDS, you must make the changes discussed in the following table to reconfigure the environment.

Table 1-1 Reconfiguration Required to Transition from an IDS Environment

Area	IDS Environment	Transition Task
Accessing the server engine	<p>IDS server and client components had to be installed and configured.</p> <p>All of the settings except for USE IDS have been removed. The Use IDS setting now just specifies file location mapping based on information in the text file <code>idshosts</code>. See "Use IDS" on page 5-45 in <i>Advanced Operations Guide</i>.</p>	<p>Ensure that Use Remote MicroKernel Engine is set to "On" (the default). See "Use Remote MicroKernel Engine" on page 5-60 in <i>Advanced Operations Guide</i>.</p> <p>The remote access method now uses the Network Services Layer (NSL) to access the server.</p>
Using encryption	<p>The IDS client and server used compression to keep the data from flowing across the network as text. Pervasive PSQL currently does not implement compression but does use encryption.</p>	<p>Enable encryption on either the client, the server or both. By default, both client and server use encryption if the other partner requires it.</p> <p>See "Wire Encryption" on page 5-13 and "Wire Encryption Level" on page 5-14, both in <i>Advanced Operations Guide</i>.</p>

Table 1-1 Reconfiguration Required to Transition from an IDS Environment *continued*

Area	IDS Environment	Transition Task
<p>Specifying user names and passwords on the client</p>	<p>IDS handled its own security involving user names and passwords with the settings IDS Username and IDS Password. Both of these setting have been removed.</p> <p>User name and password were stored as plain text. The same information would be used by all users on the IDS system. The same user name and password needed to be valid on all the IDS servers accessed.</p>	<p>The security model for Pervasive PSQL has changed. Enabling security avoids the requirement for port 139 to be open for authentication from a Windows client to a Windows server, thus allowing access over the Internet. See "Pervasive PSQL Security" on page 7-1 in <i>Advanced Operations Guide</i>.</p> <p>With security enabled, the client must provide a user name and password for database access. Select one, or a combination of, the following methods to accomplished this.</p> <ul style="list-style-type: none"> ◆ The application can specify the user name and password by using a Btrieve URI. See "Database URIs" on page 4-35 in <i>Pervasive PSQL Programmer's Guide</i>, which is part of the Pervasive PSQL software development kit (SDK). ◆ The user name and password information can be stored in the registry. See "pvnetpass" on page 8-36 in <i>Pervasive PSQL User's Guide</i>. ◆ The user can be prompted for the user name and password. See "Allow Client-stored Credentials" on page 5-7 and "Prompt for Client Credentials" on page 5-12, both in <i>Advanced Operations Guide</i>.
<p>Creating a database</p>	<p>Using the IDS server IDS Setup program, an IDS administrator could set up various database sets. For each set, the administrator could specify the path for Read Only, Write Only, and Full Access.</p>	<p>Use the PCC to create a new database. See "To create a new database" on page 1-25 in <i>Advanced Operations Guide</i>.</p> <p>Add multiple paths for the data file locations. See "Database Properties" on page 1-20 in <i>Advanced Operations Guide</i>.</p>
<p>Securing a database</p>	<p>If security was specified in the IDS Setup program, the Data Server Security program allowed Groups or Users to be added who had Read Only, Write Only or Full Access privileges.</p> <p>IDS allowed only use of operating system users and groups.</p>	<p>Security may be applied to the database through the database properties dialog. You may then add groups and users, each with specified privileges.</p> <p>Note that you may define users and groups for the database, use the operating system users and groups, or a mixture of the two.</p> <p>See "Pervasive PSQL Security" on page 7-1 in <i>Advanced Operations Guide</i>.</p>

Table 1-1 Reconfiguration Required to Transition from an IDS Environment *continued*

Area	IDS Environment	Transition Task
Configuring the network	<p>The IDS server allowed an administrator to configure the server to be multihomed. The server could listen on all IP addresses available on the system or listen on a particular IP address. Multihomed was the default.</p> <p>The IDS server listened on port 2441.</p>	<p>The Pervasive PSQL server also allows an administrator to configure the server to be multihomed. Multihomed is the default. See “Listen IP Address” on page 5-17 and “TCP/IP Multihomed” on page 5-18, both in <i>Advanced Operations Guide</i>.</p> <p>The Pervasive PSQL Communication Manager listens on port 3351. For remote access to occur, any firewalls and routers must allow access to port 3351. Port 2441 is no longer needed.</p>
Specifying an acknowledge interval	<p>The IDS server allowed an administrator to configure a time period that the server would wait before verifying that an inactive user was still present.</p>	<p>If this type of functionality is desired, you must enable AutoReconnect for both the database engine and the clients. See “Pervasive Auto-Reconnect” on page 3-21 in <i>Advanced Operations Guide</i>.</p>
Logging connection and disconnection activity	<p>The IDS server allowed for logging of connection and disconnection activity.</p>	<p>Pervasive PSQL currently does not provide the logging of connection and disconnection activity.</p>
Prompting for database set names	<p>The IDS server allowed the following:</p> <ul style="list-style-type: none"> ◆ prompting the ActiveX and Java Class Library (JCL) clients for the database name ◆ publishing the names of the configured database sets to those clients. 	<p>Pervasive PSQL currently does not provide the prompting for database set names.</p>

Application Changes

The removal of IDS may require some changes to your application. The following effects should be noted.

- IDS Name and IDS Password configuration parameters have been removed. The database engine handles all security.
- Optionally, you may want to convert IDS PIDS protocol strings. The client requester now converts PIDS string to the URI security format.
- Optionally, you may still use the idshosts file if you choose.



Note Pervasive PSQL Server 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 Pervasive PSQL 8.5.

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” on page 5-45 and “Use Remote MicroKernel Engine” on page 5-60, both in *Advanced Operations Guide*.

Security

See “Pervasive PSQL Security” on page 7-1 in *Advanced Operations Guide* for the types of database security.

The pvnetpass utility may be used to manage user ids and passwords for remote servers to which your client connects. See “pvnetpass” on page 8-36 in *Pervasive PSQL User's Guide*.

A user name and password are needed only if other authentication methods fail, or if database security is enabled and requires a database user ID and password.

Conversion of PIDS Strings

If you choose, you may convert PIDS URL strings in your applications to the BTRV URI format. Such conversion is optional because the client requester automatically converts PIDS strings to the URI format.

For example, the format of the PIDS URL is PIDS://<servername | IP address>[:<dataset>|/<filepath>]. The requester converts this to BTRV://@host/dbname?dbfile=filename.

The conversion of parameters corresponds as follows:

PIDS	BTRV
server_name or IP address	host
dataset	dbname
filepath	filename

See “Database URIs” on page 4-35 in *Pervasive PSQL Programmer's Guide*, which is part of the Pervasive PSQL software development kit (SDK).

Idshosts File

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`. You may still use the `idshosts` file if you choose by setting the “Use IDS” configuration property to “on.” Note, however, that performance is slower when the `idshosts` file is used.

Refer to the comments in the `idshosts` file itself for examples of how to map file locations. See also “Use IDS” on page 5-45 in *Advanced Operations Guide* and “Using the `idshosts` File” on page 1-19 in *Advanced Operations Guide*.

5.x Files Support Reduced

This release deprecates support for the version 5.x file format. The Pervasive PSQL v9 engine will read version 5.x format files, but will not write to them.

The previous release of Pervasive PSQL can read and modify 5.x format files. With Pervasive PSQL v9 it is no longer possible to modify 5.x format files. If you have files in 5.x format, you will have to rebuild them to a newer file format to update them.

The following effects should be noted:

- 5.x format files will be opened read-only. Any attempt to write to a 5.x format file will fail with status 46.
- Create operation will fail if the format specified is 0500 or less. The error code returned is 41 (Operation Not Allowed).
- The File Version engine setting can no longer be set to 0500.

For more information about this topic, see the following in *Advanced Operations Guide*:

- “Rebuild Utility Concepts” on page 14-2
- “Create File Version” on page 5-19

Also, refer to the following status codes:

- 41: The MicroKernel does not allow the attempted operation.
- 46: Access to the requested file is denied.

Configuration Settings Not Manually Adjustable

The following configuration settings are no longer manually adjustable. The “Change” column in the following table explains why.

Table 1-2 Configuration Settings Not Manually Adjustable

Name	Type	Change
Brouter Communication Buffer Size	Server (NetWare)	Set internally
Communication Buffer Size	Server	Set internally
Compress IDS Data	Client	IDS removed from product
IDS Password	Client	IDS removed from product
IDS Username	Client	IDS removed from product
MKDE Communication Buffer Size	Server	Set internally
Read Buffer Size	Server	Set internally
Supported Protocols	Linux server and client	Set internally
Target Engine	Client	Obsolete

Configuration Settings With Modified Defaults

The following configuration settings have modified default values.

Table 1-3 Configuration Settings With Modified Defaults

Name	Type	New Default	Refer To <i>Advanced Operations Guide</i>
Embedded Spaces	Client	On	“Embedded Spaces” on page 5-53
Splash Screen	Client and Client 16-bit	Off	“Splash Screen” on page 5-54 “Splash Screen” on page 5-60

Configuration Settings With Modified Values

The following configuration settings have modified values.

Table 1-4 Configuration Settings With Modified Values

Name	Type	Change	Refer To <i>Advanced Operations Guide</i>
Create File Version	Server	<ul style="list-style-type: none"> ◆ 9.x added ◆ 5.x removed 	“Create File Version” on page 5-19
Supported Protocols	<ul style="list-style-type: none"> ◆ Server ◆ Client ◆ Client 16-bit 	Vendor names removed from protocols	“Supported Protocols” on page 5-17 “Supported Protocols” on page 5-50 “Supported Protocols” on page 5-62

Configuration Settings Not Requiring an Engine Restart

The following server configuration settings no longer require that you restart the database engine for the setting to take effect.

Table 1-5 Configuration Settings Not Requiring an Engine Restart

Name	Refer To <i>Advanced Operations Guide</i>
Allow Client-stored Credentials	“Allow Client-stored Credentials” on page 5-7
Create File Version	“Create File Version” on page 5-19
Initiation Time Limit	“Initiation Time Limit” on page 5-21
Max MicroKernel Memory Usage	“Max MicroKernel Memory Usage” on page 5-39
Operation Bundle Limit	“Operation Bundle Limit” on page 5-21
Prompt for Client Credentials	“Prompt for Client Credentials” on page 5-12
System Data	“System Data” on page 5-19

Configuration Settings That Dynamically Increase

The configuration setting **Communication Threads** now dynamically increases up to the maximum value allowed. For Window and Linux platforms, the maximum is 1,024. For NetWare, the maximum is 200.

See “Communications Threads” on page 5-36.

Windows 16-bit Client Configuration

You now manually configure the settings for a Windows 16-bit client requester in the file BTI.INI. Windows 16-bit clients can no longer be configured through PCC.

See “Win16 Client Configuration Parameters” on page 5-59 in *Advanced Operations Guide*.

File Size Support

The support for file size now includes the following enhancements:

- “Maximum File Size of 128 Gigabytes”
- “File Segmentation”
- “Automatic Upgrade of File Version”

The following topics in *Advanced Operations Guide* provide additional details pertaining to these enhancements:

- “Pervasive PSQL Database Concepts” on page 1-2
- “File Size” on page 1-7
- “Create File Version” on page 5-19

Maximum File Size of 128 Gigabytes

In previous releases, the maximum size of a data file was 64 GB. The maximum size is now 128 GB. A new file format supports the increased file sizes. You must use the Pervasive PSQL v9 file format to have a single file size larger than 64 GB. However, you do not need to rebuild a version 8.x file to use the increased limit.

File Segmentation

In previous releases, a data file was automatically broken into 2 GB operating system file segments as its size passed that boundary. Segmentation is now optional.

A new configuration property in Pervasive PSQL Control Center, “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 size is more efficient disk I/O, which usually results in 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” on page 4-4 in *Advanced Operations Guide*.

The property is set to “Yes” by default, causing files to segment at 2 GB boundaries as with previous releases. If you set the property to “No,” version 9 files can increase past the 2 GB boundary without being segmented. See also “Automatic Upgrade of File Version” for additional information relating to the configuration property.

Any non-segmented large files are still 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.

Automatic Upgrade of File Version

If the configuration property “Create File Version” is set to 9 (the default), version 8.x files are automatically converted to version 9 files when they reach the file limits for version 8.x, which is 64 GB. The following table summarizes this behavior.

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

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 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 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 occurs. As the file grows beyond 64 GB, it will continue to segment until it reaches the maximum size allowed for a version 9 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” on page 4-4 in *Advanced Operations Guide*.

Installation

The following enhancements have been added to the installation of Pervasive PSQL:

- “Improved Installation Modularity”

Improved Installation Modularity

Previous to this release, the Pervasive PSQL engine, utilities, and documentation were installed by a single installation program.

Pervasive PSQL v9 has multiple installations:

- Engine installation

This installation installs the components necessary for the engine. There is a custom install option to exclude utilities and documentation in this install.

- Client installation

Installs all the Pervasive PSQL v9 client components and accompanying documentation.

Linux

Pervasive PSQL v9 improves Linux support in the following ways:

- “Performance Equality with Windows”
- “Client Use of idshosts File and PIDS Protocol Strings”

Performance Equality with Windows

This release increases the relational engine performance on Linux by adapting a common-address space technology used previously only on Windows.

Now the Linux engine should have performance similar to the performance on Windows, assuming similar hardware.

Client Use of idshosts File and PIDS Protocol Strings

Typically, an application provides its own file location information. As an alternative, you may provide file location mapping based on information in the text file `idshosts`. If you choose, a Linux client requester can use `idshosts` when communicating with a Pervasive PSQL server on Windows, Linux, or NetWare.

You must add the `idshosts` file to the `$PVSW_ROOT/etc` directory (by default, `/usr/local/psql/etc`). `Idshosts` may include uniform naming convention (UNC) file locations. Linux requires the use of forward slashes in UNC paths.

Legacy applications that used IDS could also pass file location information through the use of a PIDS protocol string. The Linux client requester automatically converts PIDS strings to the URI format. See “Conversion of PIDS Strings” on page 1-8.

Network Communications

As with previous releases, the client requester issues a request to named pipes to attempt to locate a database engine on the network. The Workgroup Engine now responds to such requests, not just the Windows Server Engine. From the response, a client requester can locate Workgroup Engines easier and faster.

Programming Interfaces

This release of Pervasive PSQL includes the following changes to the programming interfaces:

- “Btrieve API Improvements”
- “Distributed Tuning Interface Improvements”
- “Increased Accuracy in Extended Percentage Operations” on page 1-18
- “Delete and Rename Subfunctions for the Create Operation” on page 1-19

Btrieve API Improvements

Increased Accuracy in Extended Percentage Operations

The accuracy of `GetByPercentage` (44) and `FindPercentage` (45) operations have been improved with respect to finding a record or a percentage along a key path.

The improvements to the percentage operations for Pervasive PSQL v9 are:

- Revisions to the percentage operations implementation that improve the accuracy of the operations.
- The addition of a granularity setting in both operations that allow you to choose the factor by which the percentage is measured. In previous releases, this value was always 10,000.



Note The improved accuracy in Pervasive PSQL v9 is available whether or not you use the expanded API to specify a granularity other than 10,000.

If you do want to specify the granularity, you can do so when performing either operation by following these steps:

➤ **To specify a granularity in extended percentage operations**

- 1 Place `EXPC` in the second four bytes of the data buffer.
- 2 Place the desired granularity in the third four bytes as a LoHi Intel integer. The granularity you choose can be any number from 1 to `0xFFFFFFFF`.

3 Ensure that your data buffer length is at least 12 bytes.

For example if you want to get the 100th record from a file that contains 365 records, you can use `GetByPercentage` with 100 as the percentage, and 365 as the granularity.

Delete and Rename Subfunctions for the Create Operation

The Create operation now has two additional subfunctions that you can use to delete or rename files.

In releases prior to Pervasive PSQL v8.5, it was always possible to manipulate Btrieve files through the operating system, because Btrieve depended on the rights and privileges given by the operating system to the Btrieve user.

If you have a secure Pervasive PSQL v9 database, however, these operating system access rights may have been removed in the process of securing the database from unauthorized access. This makes programmatically deleting or moving the files difficult since the rights required are not always available.

Using the Rename and Delete Subfunctions

The rename and delete subfunctions are implemented as Create operations with new key numbers. You do not need to provide a file specification as you do when creating a new data file. The following table shows how you set up the Create operation to use the rename or delete subfunctions.

Table 1-6 Create Operation Subfunctions

Function	Key Number to Use	Description	Place in Data Buffer	Place in Key Buffer
Rename File	-127	Rename an existing file in the data buffer to the name in the key buffer	Existing File Name	New File Name
Delete File	-128	Delete a file	N/A	Existing File Name

These APIs have all been modified to work with the security model in that they will accept a URI in place of a file name in the key buffer and data buffer, if needed, to indicate a Btrieve file to delete or

rename. This allows you to provide security information with the operation.

The security information is processed just like a normal Create or Open operation. The user must be authenticated and have DB_RIGHT_CREATE, DB_RIGHT_ALTER and DB_RIGHT_OPEN privileges for the existing files and for the directory where the new file will be located if applicable.

Table 1-7 Create Subfunctions - use of URI Parameters in Key Buffer

Function	URI parameter file=	URI parameter dbfile=	URI parameter table=
Rename	✓	✓	✗
Delete	✓	✓	✗

Table 1-8 Create Subfunctions - use of URI Parameters in Data Buffer

Function	file=	dbfile=	table=
Rename	✓	✓	✗
Delete	Not applicable	Not applicable	Not applicable

Notes on Rename and Delete Subfunctions

- The previous functionality of the Create operation is intact. Follow the existing documentation on the Create operation if you want to create a new MicroKernel data file.
- The RenameFile and DeleteFile subfunctions cannot be used on files that are bound to specific databases because they do not affect the contents of the miscellaneous page.
- If a file contains an Owner name, the owner name check is not performed by the new subfunctions. The owner name is still needed to view the contents of the files.

Distributed Tuning Interface Improvements

The Distributed Tuning Interface (DTI) has these improvements in Pervasive PSQL v9:

- New APIs to better integrate with the security features introduced with Pervasive.SQL v8.5.

For more information, see:

- “PvOpenDatabase()” on page 1-22
- “PvCloseDatabase()” on page 1-23.
- “PvSecureDatabase()” on page 1-24
- “PvUnSecureDatabase()” on page 1-26
- “PvIsDatabaseSecured()” on page 1-27

As part of this addition, the PvCreateDictionary() and PvOpenDictionary() APIs are now deprecated. Future releases will not support these APIs although support is not removed for them in Pervasive PSQL v9.

- New PvCopyDatabase() API to copy an entire database to a new database, adjusting referential integrity if needed.

For more information, see “PvCopyDatabase()” on page 1-28.

- Expanded return codes for PvConnectServer() so that you can better detect the exact cause of a connection failure.

For more information, see “Enhanced PvConnectServer() Error Messages” on page 1-30.

- PvDropDatabase - a previously reserved parameter is now used as an options bitmask. Substitute the following table row for the options parameter for PvDropDatabase() in the Pervasive.SQL v8.5 SDK documentation.

In	option	Bit mask that specifies options. Set the low order bit to one (0001h) if you want data files to be deleted in addition to the database name. Otherwise, only the database name will be deleted but data and DDF files will remain.
----	--------	--

- The following restrictions have been added to PvAddTable().
 - If you use PvOpenDictionary() to obtain a dictionary handle, you will encounter an error (PCM_errFailed) if you use PvAddTable() against a secured database. Use “PvOpenDatabase()” instead and specify security credentials.
 - When describing the indexes, you must ensure that every index has segment numbers beginning with zero. When the index contain multiple segments, each segment must start numbering at zero. If no zero is found, you will encounter the error PCM_errFirstSegmentNumberMustBeZero.

PvOpenDatabase()

Opens a database by name and returns a handle that can be used to manipulate the database catalog. This function replaces PvOpenDictionary().

Header file: CATALOG.H

Requires: W3DBAV90.DLL or higher version

Syntax

```
BTI_API PvOpenDatabase (  
    BTI_LONG           hConnection,  
    BTI_CHAR_PTR      dbName,  
    BTI_CHAR_PTR      dbUser,  
    BTI_CHAR_PTR      dbPassword,  
    BTI_WORD_PTR      dbHandle) ;
```

Arguments

In	<i>hConnection</i>	Connection handle that identifies the server. Connection handles are obtained with the PvConnectServer() function.
In	<i>dbName</i>	Name of the database.
In	<i>dbUser</i>	Database user name if security is defined.
In	<i>dbPassword</i>	Database password if security is defined.
In	<i>dbHandle</i>	Returned handle to the database.

Return Values

P_OK	The operation was successful.
P_E_INVALID_HANDLE	Invalid connection handle.
P_E_NULL_PTR	Call with NULL pointer
P_E_ACCESS_RIGHT	Insufficient access right for the operation.
P_E_FAIL	Failed to open the database for other reasons.
PCM_errSessionSecurityError	Invalid user name or password

Remarks

The following preconditions must be met:

- DTI session started by calling **PvStart()**
- Connection established by **PvConnectServer()** or if you are performing the operation on a local machine, **P_LOCAL_DB_CONNECTION** may be used as the connection handle.
- If the database has security enabled, you must specify a valid database user name and password. Security for the returned database handle is enforced based on the access rights defined for the database, and should match behavior seen in SQL or ODBC interfaces.

PvCloseDatabase()

Closes an open database handle.

Header file: **CATALOG.H**

Requires: **W3DBAV90.DLL** or higher version

Syntax

```
PRESULT PvCloseDatabase (
    PS_UINT16      dbHandle) ;
```

Arguments

In	<i>dbHandle</i>	Handle to a database opened by PvOpenDatabase() [page 1-22].
----	-----------------	---

Return Values

PCM_Success	The operation was successful.
PCM_errFailed	Operation was not successful.
PCM_errMemoryAllocation	An error occurred during memory allocation
PCM_errDictionaryNotOpen	No database open with specified handle.

Remarks

The following preconditions must be met:

- DTI session started by calling **PvStart()**
- Connection established by **PvConnectServer()** or if you are performing the operation on a local machine, **P_LOCAL_DB_CONNECTION** may be used as the connection handle.
- Valid database handle returned by **PvOpenDatabase()** on page 1-22.

PvSecureDatabase()

Enables database security for an existing database.

Header file: **CATALOG.H**

Requires: **W3DBAV90.DLL** or higher version

Syntax

```
BTI_API PvSecureDatabase (
    BTI_LONG          hConnection,
    BTI_CHAR_PTR      dbName,
    BTI_CHAR_PTR      dbUser,
    BTI_CHAR_PTR      dbPassword) ;
```

Arguments

In	<i>hConnection</i>	Connection handle that identifies the server. Connection handles are obtained with the PvConnectServer() function.
In	<i>dbName</i>	Name of the database.
In	<i>dbUser</i>	Database user name - must be Master to set security.

In	<i>dbPassword</i>	Database password to use for Master user.
----	-------------------	---

Return Values

P_OK	The operation was successful.
P_E_INVALID_HANDLE	Invalid connection handle.
P_E_NULL_PTR	Call with NULL pointer
P_E_ACCESS_RIGHT	Insufficient access right for the operation.
P_E_FAIL	Failed to open the database for other reasons.
PCM_errSessionSecurityError	Invalid user name or password

Remarks

The following preconditions must be met:

- DTI session started by calling **PvStart()**
- Connection established by **PvConnectServer()** or if you are performing the operation on a local machine, **P_LOCAL_DB_CONNECTION** may be used as the connection handle.
- When you enable database security, you must specify Master as the database user name and choose a password. Security for the database is enforced based on the access rights defined for the database, and should match behavior seen in SQL or ODBC interfaces.

PvUnSecureDatabase()

Disables database security on a database.

Header file: CATALOG.H

Requires: W3DBAV90.DLL or higher version

Syntax

```
BTI_API PvUnSecureDatabase (  
    BTI_LONG                hConnection,  
    BTI_CHAR_PTR            dbName,  
    BTI_CHAR_PTR            dbUser,  
    BTI_CHAR_PTR            dbPassword) ;
```

Arguments

In	<i>hConnection</i>	Connection handle that identifies the server. Connection handles are obtained with the PvConnectServer() function.
In	<i>dbName</i>	Name of the database.
In	<i>dbUser</i>	Database user name - must be Master to enable or disable security.
In	<i>dbPassword</i>	Database password for Master user.

Return Values

P_OK	The operation was successful.
P_E_INVALID_HANDLE	Invalid connection handle.
P_E_NULL_PTR	Call with NULL pointer
P_E_ACCESS_RIGHT	Insufficient access right for the operation.
P_E_FAIL	Failed to open the database for other reasons.
PCM_errSessionSecurityError	Invalid user name or password

Remarks

The following preconditions must be met:

- DTI session started by calling **PvStart()**.

- Connection established by **PvConnectServer()** or if you are performing the operation on a local machine, **P_LOCAL_DB_CONNECTION** may be used as the connection handle.
- Database is secured

PvIsDatabaseSecured()

Determines whether a given database has security enabled.

Header file: **CATALOG.H**

Requires: **W3DBAV90.DLL** or higher version

Syntax

```
BTI_API PvIsDatabaseSecured(
    BTI_LONG          hConnection,
    BTI_CHAR_PTR     dbName,
    BTI_LONG_PTR     secured);
```

Arguments

In	<i>hConnection</i>	Connection handle that identifies the server. Connection handles are obtained with the PvConnectServer() function.
In	<i>dbName</i>	Name of the database to check.
Out	<i>secured</i>	1 if database is secured 0 if database is not secure

Return Values

P_OK	The operation was successful.
P_E_INVALID_HANDLE	Invalid connection handle.
P_E_NULL_PTR	Call with NULL pointer
P_E_ACCESS_RIGHT	Insufficient access right for the operation.
P_E_FAIL	Failed to open the database for other reasons.

Remarks

The following preconditions must be met:

- DTI session started by calling **PvStart()**
- Connection established by **PvConnectServer()** or if you are performing the operation on a local machine, **P_LOCAL_DB_CONNECTION** may be used as the connection handle.

PvCopyDatabase()

Copies a database to a new database, adjusting the referential integrity of needed.

Header file: **CATALOG.H**

Requires: **W3DBAV90.DLL** or higher version

Syntax

```
BTI_API PvCopyDatabase (  
    BTI_LONG          hConnection,  
    BTI_CHAR_PTR     dbName,  
    BTI_CHAR_PTR     newdbName,  
    BTI_CHAR_PTR     newdictPath,  
    BTI_CHAR_PTR     newdatapath);
```

Arguments

In	<i>hConnection</i>	Connection handle that identifies the server. Connection handles are obtained with the PvConnectServer() function.
In	<i>dbName</i>	Name of the database to copy.
In	<i>newdbName</i>	Name of the new database.
In	<i>newdictPath</i>	Dictionary path of the new database. .
In	<i>newdatapath</i>	Data path. Pass an empty string to use the default data path (that is, the same as the dictionary path) If you want to create a new database that consists of MicroKernel data files located in multiple paths, specify this parameter as a semicolon (;) delimited list. For example: C:\data\path1;C:\data\path2

Return Values

P_OK	The operation was successful.
P_E_INVALID_HANDLE	Invalid connection handle.
P_E_NULL_PTR	Call with NULL pointer
P_E_ACCESS_RIGHT	Insufficient access right for the operation
P_E_DICTIONARY_ALREADY_EXISTS	Cannot create dictionary because it already exists.
P_E_SHARED_DDF_EXIST	Cannot create DDF files because
P_E_DUPLICATE_NAME	Named database already exists on the server.
P_E_FAIL	Failed for other reasons.

Remarks

The following preconditions must be met:

- DTI session started by calling **PvStart()**.
- Connection established by **PvConnectServer()** or if you are performing the operation on a local machine, **P_LOCAL_DB_CONNECTION** may be used as the connection handle.

Example

```
BTI_LONG connectionHandle = P_LOCAL_DB_CONNECTION;
BTI_CHAR_PTR newdataPath = "c:\\data\\gallery2";
BTI_CHAR_PTR newdictPath = "c:\\data\\gallery2";
BTI_CHAR_PTR databaseName = "Gallery";
BTI_CHAR_PTR newdatabaseName = "GalleryCopy";
BTI_SINT status = 0;
BTI_CHAR_PTR server = "MyServer";
BTI_CHAR_PTR user = "Administrator";
BTI_CHAR_PTR pwd = "Admin";
//only need to connect to server if it is remote
//otherwise can pass P_LOCAL_DB_CONNECTION for the
handle

status = PvCopyDatabase(
connectionHandle,
databaseName,
newdatabaseName
```

```
dictPath,  
dataPath);
```

Enhanced PvConnectServer() Error Messages

Prior to this release, the `PvConnect()` API returned the error message `P_E_FAIL` (connection failed) for all connection-related failures. The true cause of the connection failure was not easily ascertained.

In this release, `PvConnect()` will return any of the following status codes in place of `P_E_FAIL` (status 7004).

Table 1-9 New Error Messages for PvConnectServer()

New Code	Explanation
<code>P_E_SERVER_NOT_FOUND</code>	The specified server was not found
<code>P_E_ENGINE_NOT_LOADED</code>	The specified engine is not running.
<code>P_E_REQUESTER_NOT_LOADED</code>	The client requester is not loaded.
<code>P_E_SERVER_TABLE_FULL</code>	The internal server name table is full.
<code>P_E_CLIENT_CONNECTIONS_LIMIT_REACHED</code>	The operation could not connect because the limit on client connections has been reached. Check the configuration of the server.
<code>P_E_PERMISSION_ERROR</code>	The operation encountered a permissions error.
<code>P_E_NO_MEMORY</code>	The operation encountered a memory error.
<code>P_E_NO_AVAILABLE_TRANSPORT</code>	No remote connection could be established.
<code>P_E_CONNECTION_LOST</code>	The remote connection to the server was lost.

Relational Interface Support

This section describes the new or revised functionality to support the relational interface.

- “ALTER (rename)” on page 1-31
- “ALTER TABLE Enhancements” on page 1-31
- “Bitwise Operators” on page 1-32
- “CASE Expression” on page 1-32
- “DEFAULT Keyword” on page 1-32
- “EXECUTE Statement” on page 1-33
- “FROM Clause in UPDATE and DELETE Statements” on page 1-33
- “Global Variables” on page 1-33
- “Grouped View Support” on page 1-34
- “Linked Duplicate Keys for CREATE TABLE” on page 1-34
- “ODBC 3.5 Support” on page 1-34
- “Promotion of Numeric Data Types” on page 1-34
- “Scalar Functions” on page 1-35
- “SELECT Statement” on page 1-36
- “SET ANSI_PADDING” on page 1-36
- “Subquery Expressions” on page 1-36
- “System Stored Procedures” on page 1-37
- “User Defined Functions” on page 1-37

ALTER (rename)

The ALTER statement allows you to change the name of indexes, user-defined functions, stored procedures, tables, triggers, or views. See “ALTER (rename)” on page 3-5 in *SQL Engine Reference*.

ALTER TABLE Enhancements

ALTER TABLE now provides syntax to rename a column and to move a column.

Column Rename

New syntax allows you to change the name a column from its existing name to a new name.

Column Move

New syntax allows you to move table columns logically or physically to keep columns at desired ordinal positions. You may also change the ordinal position of a new column after adding it.

For both enhancements, see “ALTER TABLE” on page 3-10 in *SQL Engine Reference*.

Bitwise Operators

Bitwise operators allow you to manipulate the bits of one or more operands. The following bitwise operators are supported:

Operator	Meaning
&	bitwise AND
~	bitwise NOT
	bitwise OR
^	bitwise exclusive OR

See “Bitwise Operators” on page 4-1 in *SQL Engine Reference*.

CASE Expression

A CASE expression returns a value. CASE expression has two formats:

- Simple When/Then. This format compares a value expression to a set of value expressions to determine a result. The value expressions are evaluated in their order listed. If a value expression evaluates to TRUE, CASE returns the value expression for the THEN clause.
- Searched When/Then. This format evaluates a set of Boolean expressions to determine a result. The Boolean expressions are evaluated in their order listed. If a Boolean expression evaluates to TRUE, CASE returns the expression for the THEN clause.

See “CASE (expression)” on page 3-33 in *SQL Engine Reference*.

DEFAULT Keyword

You may use the DEFAULT keyword in the following contexts:

- Column definition of CREATE TABLE statement
- Column definition of ALTER TABLE statement
- VALUES clause of INSERT statement
- VALUES clause of UPDATE statement

The default value (literal or expression) that you specify in a CREATE TABLE or ALTER TABLE statement must meet the following criteria:

- be consistent with the data type of the column
- conform to any other constraint imposed on the column, such as range, length, and so forth.

In INSERT and UPDATE statements, you do not have to specify values for columns that have a DEFAULT value defined. In such a case, Pervasive PSQL computes the DEFAULT expression and uses the result value as the value of the column.

See “DEFAULT” on page 3-104 in *SQL Engine Reference*.

EXECUTE Statement

The EXECUTE statement has two uses:

- To invoke a user-defined procedure or a system stored procedure. You may use EXECUTE in place of the CALL statement
- To execute a character string, or an expression that returns a character string, within a stored procedure.

See “EXECUTE” on page 3-126 in *SQL Engine Reference*.

FROM Clause in UPDATE and DELETE Statements

The UPDATE and DELETE statements, when used at the session level, now support a FROM clause containing a table reference. The FROM clause does not apply to the following:

- UPDATE or DELETE statements used in a stored procedure
- the positioned UPDATE statement
- the positioned DELETE statement.

See “DELETE” on page 3-112 and “UPDATE” on page 3-243, both in *SQL Engine Reference*.

Global Variables

The following global variable has been added:

- @@SPID Variable

@@SPID

This variable (server process identifier) returns the identifier integer value of the system thread for the Pervasive PSQL connection.

See “@@SPID” on page 3-266 in *SQL Engine Reference*.

Grouped View Support

A grouped view is one that contains any of the following in the SELECT list:

- DISTINCT
- GROUP BY
- Scalar Functions
- Scalar subquery
- TOP
- UNION

Grouped views may be used in a subquery provided the subquery is an expression. A subquery connected with the operators IN, EXISTS, ALL, or ANY is **not** considered an expression.

See “CREATE VIEW” on page 3-99 in *SQL Engine Reference*.

Linked Duplicate Keys for CREATE TABLE

The CREATE TABLE statement provides a new keyword, LINKDUP, that allows you to specify the number of pointers to reserve for the addition of linked duplicate index 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.)

See the following topics:

- “LINKDUP” on page 3-80 in *SQL Engine Reference*
- “Methods for Handling Duplicate Keys” on page 13-13 in *Advanced Operations Guide*

ODBC 3.5 Support

The Pervasive relational interface fully conforms to the ODBC v3.51 specifications for Core, Level 1, and Level 2 interface conformance levels.

See “ODBC Support” on page 2-8 in *SQL Engine Reference*.

Promotion of Numeric Data Types

The following topics pertain to the promotion of numeric data types:

- Operator Precedence
- Data Type Precedence
- Precision and Scale of Decimal Data Types

Operator Precedence

An expression may have multiple operators. Operator precedence determines the sequence in which the operations are performed. An operator on a higher level is evaluated before an operator on a lower level.

See “Operator Precedence” on page A-6 in *SQL Engine Reference*.

Data Type Precedence

Data type precedence determines which data type results when two expressions of different data types are combined by an operator. The data type with the lower precedence is converted to the data type with the higher precedence.

See “Data Type Precedence” on page A-8 in *SQL Engine Reference*.

Precision and Scale of Decimal Data Types

Precision is the number of digits in a number. Scale is the number of digits to the right of the decimal point in a number. The number 909.777 has a precision of 6 and a scale of 3, for instance.

See “Precision and Scale of Decimal Data Types” on page A-9 in *SQL Engine Reference*.

Scalar Functions

The following scalar functions have been added:

- CAST
- COALESCE
- ISNULL

CAST

The CAST function can convert data to any Pervasive PSQL relational data type. The CAST function does **not** support converting binary zeros in a string. For example, the following is invalid: CAST(c1 AS BINARY(10)), where c1 is a character column that contains binary zeros.

You cannot CAST user-defined data types. If both the input and the output are character strings, the output from CAST has the same collation as the input string.

See “Conversion Functions” on page 5-20 in *SQL Engine Reference*.

COALESCE

This function takes two or more arguments and returns the first nonnull argument, starting from the left in the expression list.

See “Logical Functions” on page 5-17 in *SQL Engine Reference*

ISNULL

This function takes two arguments (*exp* and *value*) and replaces NULL with the value specified for the *value* argument. *Exp* is the expression to check for NULL. *Value* is the value returned if *exp* is NULL. *Exp* is returned if it is not NULL. The data type of *value* must be compatible with the data type of *exp*.

See “Logical Functions” on page 5-17 in *SQL Engine Reference*

SELECT Statement

The SELECT statement now supports the following:

- CASE as expression
- COALESCE as expression
- NULL as expression
- Subquery as expression
- UNION in subquery
- Column names in ORDER BY with UNION

See “SELECT” on page 3-180 in *SQL Engine Reference*.

SET ANSI_PADDING

The SET ANSI_PADDING statement allows the relational interface to handle CHAR data types padded with NULLs (binary zeros). CHAR is defined as a character data type of fixed length.

See “SET ANSI_PADDING” on page 3-206 in *SQL Engine Reference*.

Subquery Expressions

A subquery is a SELECT statement with one or more SELECT statements within it. The maximum number of nested subqueries allowed within the topmost SELECT statement is 16.

The following types of subqueries are supported:

- comparison
- quantified
- in
- exists

- correlated
- expression

See “Subqueries” on page 3-185 in *SQL Engine Reference*.

System Stored Procedures

System stored procedures help you accomplish those administrative and informative tasks that are not covered by the Data Definition Language. The system stored procedures have a “psp_” prefix.

An error results if you execute a system stored procedure in the context of a database (for example, A) and try to obtain information from a secured database (for example, B). Information from a secured database cannot be obtained by another database.

See “System Stored Procedures” on page 6-1 in *SQL Engine Reference*.

User Defined Functions

The CREATE FUNCTION statement creates a scalar user-defined function (UDF) in the database. You can invoke user-defined functions from a query.

See “CREATE FUNCTION” on page 3-45 in *SQL Engine Reference*.

Utilities and Documentation

This release of Pervasive PSQL includes the following changes to utilities and documentation:

- “Pervasive PSQL Control Center” on page 1-38
- “Command Line Interface Utilities” on page 1-38
- “JavaHelp Documentation” on page 1-39

Pervasive PSQL Control Center

This release adds a new set of Java-based GUI utilities that you can use on Windows platforms.

The core utility is Pervasive PSQL Control Center (PCC), a framework that replaces and extends functionality found in the following utilities of previous releases:

- Pervasive Control Center (replaced by Pervasive PSQL Control Center)
- Configuration Utility (replaced by properties within PCC)
- SQL Data Manager (replaced by SQL Editor)
- Table Designer (replaced by Table Editor)

Other utilities are not yet migrated to Java, but can be loaded as external tools. These include:

- Function Executor
- License Administrator
- Monitor
- Maintenance
- Rebuild
- Export



Note PCC now identifies a database solely by the internal Database Name (DBNAME). Data Source Names (DSNs) do not have to be associated with a database unless you want to access the database through ODBC.

To learn about the new PCC, see the chapter “Using Pervasive PSQL Control Center” on page 3-1 in *Pervasive PSQL User's Guide*.

Command Line Interface Utilities

This release of Pervasive PSQL includes the following additional command line interface (CLI) utilities:

- Configuration Properties: “bcfg”
- Bulk Data Utility: “bdu”
- Monitor Utility: “bmon”
- SQL Utility: “pvddl”

bcfg

The Configuration properties Utility (bcfg) is a command-line utility that allows you to change your configuration.

To learn about bcfg, see the chapter “Configuration Through CLI Utility” on page 4-5 in *Advanced Operations Guide*.

bdu

The Bulk Data Utility (bdu) is a command-line utility that allows you to load data from a delimited text file into a preexisting Pervasive PSQL table. BDU replaces the Import utility that was available with previous releases.

To learn about bdu, see “bdu” on page 8-7 in *Pervasive PSQL User's Guide*.

bmon

The Monitor Utility (bmon) is a command-line utility that allows you to monitor database resources.

To learn about bmon, see the chapter “Command Line Interface Monitor” on page 11-21 in *Advanced Operations Guide*.

pvddl

The Pervasive Data Definition Language (pvddl) utility allows you to execute one or more SQL statements from a command line interface.

To learn about pvddl, see “pvddl” on page 8-34 in *Pervasive PSQL User's Guide*.

JavaHelp Documentation

JavaHelp replaces the HTML Help functionality found in previous releases in order to support the new Java-based Pervasive PSQL Control Center. This new help functions in much the same way as

the previous HTML Help system. You can also customize the helpset to fit your needs.

JavaHelp is based on the concept of a helpset. A helpset is a definition of content, features and configuration that control the functionality of the documentation.

The previous help format (Microsoft HTML Help) is still installed by the PSA utility. This duplicate help format is installed so that it can support the utilities that have not yet been converted to the new Java platform for Pervasive PSQL Control Center. The content of this help system is identical to that of JavaHelp.

Structure of Pervasive JavaHelp

Pervasive JavaHelp is packaged in the same way and installed to the same folder location as the Pervasive PSQL Control Center.

Platform	Installation Location
Windows	C:\pvs\bin\plugins
Linux	/usr/local/psql/bin/plugins

The structure of the Pervasive help system is as follows:

Help component	Package name in \bin\plugins	Description
Pervasive Help User Interface	com.pervasive.help.ui_1.0.0	<ul style="list-style-type: none">◆ Loads the Pervasive JavaHelp window.◆ Controls the interface between PCC and the help system.
Pervasive PSQL documentation	com.pervasive.help.docs.psql.enus	Pervasive PSQL v9 documentation content.
Other Pervasive Software products	com.pervasive.help.docs.*	When you install another Pervasive Software product, the help will merge with that of Pervasive PSQL to form an integrated helpset.

Customizing the Help System

The help system can be customized by editing the file `preferences.ini` located in the package `com.pervasive.help.ui_1.0.0`. You must close and restart Pervasive PSQL Control Center and other open JavaHelp windows for the changes to `preferences.ini` to take effect. The following documentation settings can be customized:

preferences.ini setting	Possible values	Description
title	Any text. Default is 'Pervasive Library'	Controls the title of the JavaHelp window.
helpset	<ul style="list-style-type: none"> ◆ psqldocs (default) ◆ pvswdocs (ALL Pervasive products) ◆ book/folder 	<p>This setting controls which helpset is the master helpset for the Pervasive platform.</p> <p>psqldocs includes Pervasive PSQL and companion products such as DDF Builder. It does not contain SDK docs. This is the default for Pervasive PSQL v9.</p> <p>pvswdocs includes all Pervasive documentation. As products are installed, they will merge with this master helpset. pvswdocs also includes the Pervasive PSQL SDK docs and a Glossary tab. Initial load time of the help system increases if you use pvswdocs because more content must be loaded.</p>
helpset_plugins	com.pervasive.help.docs.*	Controls which plugins are scanned by JavaHelp during initial load. It is not recommended that you alter this setting.
default_topic	pvswdocs.welcome	Controls which topic in the helpset is loaded when the JavaHelp window is first displayed.
height	Any value in pixels	Height of JavaHelp window when first loaded.

preferences.ini setting	Possible values	Description
width	Any value in pixels	Width of JavaHelp window when first loaded.
locationX	Horizontal pixel coordinate from upper left	Position of JavaHelp window when first loaded.
locationY	Vertical coordinate from upper left	Position of JavaHelp window when first loaded.

Configuration of JavaHelp

Pervasive JavaHelp assumes that the Java compiler is available in the PATH after installation of JRE. If JavaHelp is not loading, ensure that your environment is set correctly:

- If JAVA_HOME environment variable is set, Pervasive JavaHelp assumes that the java executable is at <JAVA_HOME>\bin\java. JAVA_HOME is not set by the JRE install. Other vendors require a JAVA_HOME variable so it may be set on your system. If it is, then it takes precedence over any PATH statements. For example:

```

JAVA_HOME=c:\jre
JAVA_HOME=/usr/local/java
    
```

- Without a JAVA_HOME, Pervasive JavaHelp assumes that the PATH environment variable contains the location of your JRE's bin directory. For example:

```

PATH=c:\jre\bin
PATH=/usr/local/java/bin
    
```

External HTML Links in JavaHelp

JavaHelp uses a Java browser to display help content. Since the Java browser is designed for compactness, it is not optimized for the wide variety of web components found in Web sites. Thus, external links (such as <http://www.pervasive.com>) are routed to a web browser on your machine.

In Windows, the default browser Internet Explorer is used. Pervasive JavaHelp does not attempt to find where Internet Explorer is installed, so you must ensure that Internet Explorer (iexplore.exe) is in the PATH. On most Windows machines, this is already the case.

In Linux, the `htmlview` script is used to find the default browser. This script is in the public domain, and may be part of your Linux distribution. For example, it is located in `/usr/bin` on Red Hat distributions.

Index

Symbols

@@SPID 1-33

Numerics

16-bit clients

 configuring 1-11

2 GB file segments

 configuration property 1-13

A

ALTER (rename) 1-31

ALTER TABLE

 move a column 1-31

 rename a column 1-31

B

Bitwise operators 1-32

Bulk Data utility 1-39

C

CASE as expression 1-36

CASE expression 1-32

CAST scalar function 1-35

Client requester

 configuring Windows 16-bit 1-11

Clients

 configuring Windows 16-bit 1-11

COALESCE as expression 1-36

COALESCE scalar function 1-36

COBOL

 DBCobolNumeric setting 1-3

 occurs support 1-3

 partial redefines support 1-3

 sign value for positive NUMERIC data types 1-3

 SQL enhanced access 1-3

 variable record layouts 1-3

Column names in ORDER BY with UNION 1-36

Control Center

 Pervasive PSQL 1-38, 1-39

CREATE FUNCTION 1-37

D

Data type precedence 1-35

DBCobolNumeric setting 1-3

DECIMAL data types

 precision 1-35

 scale 1-35

DEFAULT keyword 1-32

DELETE

 FROM clause 1-33

E

EXECUTE statement 1-33

Expressions

 data type precedence 1-35

 operator precedence 1-35

 subquery 1-36

F

Features

 list of new and improved 1-2

File

 segmentation 1-13

 size

 limiting to 2 GB 1-13

 support for 128 GB 1-13

 version

 automatic upgrade 1-14

File version

 automatic upgrade 1-14

FROM clause in UPDATE and DELETE statements
1-33

Functions

 user defined 1-37

G

Global variables

 @@SPID 1-33

Grouped view

 in SELECT list 1-34

I

- I*net data server 1-4
 - conversion of PIDS string 1-8
 - environment configuration 1-5
 - idshosts file 1-9
 - possible application changes to use 1-7
- Idshosts file
 - configuring for Linux 1-16
 - use IDS configuration property 1-9

Import data

See Bulk Data utility

Importing data

See Bulk Data utility

Improvements

in current release 1-2

ISNULL scalar function 1-36

K

Keys

duplicate 1-34

L

LINKDUP keyword 1-34

Linked duplicate keys

for CREATE TABLE 1-34

N

New features 1-2

NULL as expression 1-36

NUMERIC

sign value for positive data type 1-3

NUMERIC data type

shifted and unshifted sign value for 1-3

NUMERIC data types

promotion of 1-34

O

ODBC

conformance 1-34
support for 3.5 1-34

ODBC 3.5

support for 1-34

Operator precedence 1-35

ORDER BY with UNION

column names in 1-36

P

PCC 1-38, 1-39

Pervasive PSQL Control Center 1-38, 1-39

PIDS strings

conversion of 1-8

Precedence

data type 1-35

operator 1-35

Precision

of DECIMAL data types 1-35

Procedures

system stored 1-37

PvCloseDatabase() 1-23, 1-27

PvConnectServer()

error messages 1-30

PvCreateDatabase() 1-26, 1-28

PvOpenDatabase() 1-22, 1-24

R

Redefines

See COBOL

S

Scalar functions 1-35

CAST 1-35

COALESCE 1-36

ISNULL 1-36

Scale

of DECIMAL data types 1-35

Segment

file segment property 1-13

SET ANSI_PADDING 1-36

Shift

value for positive NUMERIC data types 1-3

Sign value

for positive NUMERIC data types 1-3

shifted and unshifted for NUMERIC data type 1-

3

SQL

enhanced access for COBOL 1-3

Subquery as expression 1-36

Subquery expressions 1-36

System procedures 1-37

System stored procedures 1-37

U

UNION

 column names in ORDER BY 1-36

UNION in subquery 1-36

UPDATE

 FROM clause 1-33

User defined functions 1-37

User-defined functions 1-37

W

What is new in the current release 1-2

