

# Pervasive PSQL v11

---

## *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](mailto: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 2010 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](mailto:pharvey@codebydesign.com)), modified and extended by Nick Gorham ([nick@easysoft.com](mailto: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](http://www.unixodbc.org). For further information on this project, contact its current maintainer: Nick Gorham ([nick@easysoft.com](mailto: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/licenses/lgpl.html](http://www.fsf.org/licenses/lgpl.html).

### **What's New in Pervasive PSQL**

**Beta Release April 2010**

**138-004435-001**

# Contents

<b>About This Manual . . . . .</b>	<b>v</b>
Who Should Read This Manual . . . . .	vi
Manual Organization . . . . .	vii
Conventions . . . . .	viii
<b>1 What Is New in Pervasive PSQL v11 . . . . .</b>	<b>1-1</b>
<i>An Overview of New and Changed Features</i>	
Multi-core Support . . . . .	1-2
Why Multi-core Support . . . . .	1-2
The Multi-core Environment . . . . .	1-4
Benefiting from the Present While Planning the Future . . . . .	1-6
Relational Interface Support for 64-bit Architecture. . . . .	1-7
Windows Registry . . . . .	1-7
ODBC Interface Support for 64-bit Architecture. . . . .	1-9
ODBC and Data Source Names (DSNs) . . . . .	1-9
Utilities Affected . . . . .	1-14
Product Activation . . . . .	1-16
Telephone Activation . . . . .	1-16
Product Activation for OEMs . . . . .	1-16
Configuration Settings . . . . .	1-17
Utility Changes . . . . .	1-18
Pervasive PSQL Control Center. . . . .	1-18
ODBC Administrator . . . . .	1-18
Deprecated and Discontinued Features . . . . .	1-19
Deprecated Features . . . . .	1-19
Discontinued Features . . . . .	1-19

# *Tables*

1-1	Pervasive PSQL ODBC Drivers for Windows. . . . .	1-9
1-2	FAQs About ODBC and DSN Changes . . . . .	1-10

# *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 v11.

---

## 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 the Community Forum on the Pervasive Software Web site.

---

## 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 Is New in Pervasive PSQL v11”

This chapter provides an overview of the changes in the current 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 <code>MYPROG</code> , <code>myprog</code> , or <code>MYprog</code> .
<b>Bold</b>	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 <code>[log_name]</code> . If information is not enclosed in square brackets, it is required.
	A vertical bar indicates a choice of information to enter, as in <code>[file_name   @file_name]</code> .
< >	Angle brackets enclose multiple choices for a required item, as in <code>/D=&lt;5   6   7&gt;</code> .
<i>variable</i>	Words appearing in italics are variables that you must replace with appropriate values, as in <code>file_name</code> .
...	An ellipsis following information indicates you can repeat the information more than one time, as in <code>[parameter ...]</code> .
::=	The symbol <code>::=</code> means one item is defined in terms of another. For example, <code>a::=b</code> means the item <code>a</code> is defined in terms of <code>b</code> .
%string%	A variable defined by the Windows operating system. <i>String</i> represents the variable text. Example: <code>%ProgramFiles%</code> is a variable for the location <code>C:\Program Files</code> .
\$string	An environment variable defined by the Linux operating system. <i>String</i> represents the variable text. Example: <code>\$PATH</code> , which contains a colon-separated list of directories that the shell searches for commands that do not contain a slash in their name.

# *What Is New in Pervasive PSQL v11*

---

## *An Overview of New and Changed Features*

The Beta release includes the following new features and changes:

- “Multi-core Support”
- “Relational Interface Support for 64-bit Architecture”
- “ODBC Interface Support for 64-bit Architecture”
- “Product Activation”
- “Configuration Settings”
- “Utility Changes”
- “Deprecated and Discontinued Features”

## Multi-core Support

Pervasive PSQL v11 is specifically designed to increase scalability and performance of client/server applications in multi-core environments. The enhancements are predominately transparent. Install Pervasive PSQL v11 on a multi-core machine and the benefits are immediately available to your application.

You may wonder “what benefits?” Increased scalability and performance are obviously desirable. But since software applications can run on multi-core machines as well as single-core machines, will not the latest machines and operating systems provide the same benefits?

This section answers that question, and explains why multi-core support is a primary feature of Pervasive PSQL v11.

### **Why Multi-core Support**

Without modifications, almost all software applications can run on multi-core machines. But consider the following scenario, which is based on real-world feedback:

You replace your antiquated production server with a current one. Your client/server application gets installed on the new multi-core machine with a compatible operating system. Things should be humming better than ever. But response time is slower. Performance is worse than before the hardware upgrade.

What happened? Critical components of your business solution are no longer optimized for one another in the new world of multi-core.

Think of it this way. Your “application” comprises four main pieces: the code you wrote (application in its common definition), the database, the operating system, and the hardware. Changing two of these primary components—the operating system and the hardware—has a significant impact if those two components fundamentally differ from their predecessors. Those differences were at odds with the performance and scalability of Pervasive PSQL, so we greatly improved the database engine.

### **Performance**

Pervasive PSQL v11 has been architected to provide parallel threads performing similar activities. The gains in increased parallel processing improve the throughput to the point that multiple

processors are engaged. The result is that performance of the database engine *increases* in multi-core environments with multiple clients accessing a central server. Your multi-client application can benefit from this increased performance without requiring you to re-compile or re-architect the code.

Pervasive PSQL v11 also provides enhancements to the low-level synchronizations mechanisms in the transactional interface. Multiple users can read the same cached file pages simultaneously and their operations can proceed on independent server CPUs. Non-user activity such as checkpoints and log management can also use additional server CPUs.

### **Scalability**

The scalability of Pervasive PSQL v11 has also been enhanced. Many bottlenecks in the database engine have been removed or diminished. For example, multiple users accessing independent files can proceed on independent server CPUs. The database engine can also handle higher user loads with less overhead, resulting in steadier throughput.

Just as with the performance improvements, all of the scalability enhancements are available without requiring you to re-compile or re-architect your code.

### **Configuration Settings**

Most multi-core improvements in Pervasive PSQL v11 are transparent. You are not required to adjust any settings to further enhance the optimizations. The following configuration settings have changed and can be used to to fine tune performance if you choose.

#### ***Communications Threads***

The range and default for the Communications Threads setting have changed. The range is now *num\_cores* to 256, where *num\_cores* is the number of processors in the machine on which the database engine is running. The default is *num\_cores*. (Previously, the range was 1 to 1,024 and the default was 16.)

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. In Pervasive PSQL v11, worker threads are dynamically created only if all the existing threads are waiting on record or file locks.

See “Communications Threads” in *Advanced Operations Guide*.

## **The Multi-core Environment**

Just as high tide lifts all boats, the latest hardware and operating systems have traditionally boosted performance for software applications. In general, the boost required little or no effort. Applications just ran faster. The expectation of increased performance continues now that multi-core machines and 64-bit operating systems are the norm. But the expectation is no longer valid.

Several issues are at play in the multi-core world of hardware and software interaction that may cause *decreased* performance with your application:

- “Memory Contention”
- “Multiple Threads”
- “Clock Speeds”

### **Memory Contention**

When most applications were written, developers did not have to decide between parallel and non-parallel processes. The majority of applications were written sequentially, meaning that they access information serially or sequentially. A problem with memory contention occurs when running a non-parallel (typical) application on a multi-core system.

Consider the slapstick comedy skit that depicts a group of people trying to get through a single doorway at the same time. This is good for laughs because the individuals just jam together at the opening, wedged into an immovable mass. Now, image that, instead of people and a doorway, it is multiple threads trying to be processed at the same time. With four to sixteen threads (or more) trying to get through the same processor at once, a jam occurs that the operating system must sort out.

Contention also occurs as the processors repeatedly check the cache to ensure a task on one processor does not execute on outdated data produced by another task on another processor. This checking slows processing because each processor checks the memory cache individually and sequentially.

### ***False Sharing***

One of the advantages of multithreaded programming is that multiple tasks can run simultaneously. For multiple processing to run efficiently, the tasks need to establish prioritization of cache queues as they seek data from memory. Otherwise, multiple tasks that are processing entirely different data may contend for the same memory cache. This false sharing of memory pulls resources back and forth in a time-consuming struggle. To take advantage of multi-core processing, applications must be written so that such false sharing is avoided.

### **Multiple Threads**

A multithreaded application does not necessarily run better on a multi-core machine. There are a couple of reasons why.

To work efficiently in parallel, the threads must be synchronized. An application can be multithreaded, but the threads themselves not synchronized. This situation is actually quite common, in which older applications spin off additional threads as needed, more for convenience than based on a design to ensure synchronization. Such applications do not run better on a multi-core machines because the threads contend with one another. Multiple cores provide no benefit because thread contention inhibits throughput to the point that multiple cores are not engaged.

Also, the multi-core architecture can perceive the subtasks that spin off the multiple thread as a series of single threads. And, just as with single-threaded programs, the threads are then forced into a single queue and processed one by one.

### **Clock Speeds**

Because of physical properties and technological reasons, CPUs have reached their maximum speeds. If the chips cannot go faster, then how can hardware provide additional value? The answer lies in collecting multiple processors (or “cores”) into a single integrated

circuit. The collection offers more processing power as a unit, but each processor may be no faster than the previous single-core generation. In some cases, the processor is *slower* because of design considerations on older multi-core machines, such as smaller cache size.

***Benefiting from  
the Present  
While Planning  
the Future***

Recall that the “application” consists of your code, the database, the operating system, and hardware. Hardware systems have already addressed multi-core support. Multi-core machines are the norm, so any current or future hardware upgrades will include multiple cores. Operating systems have yet to catch up with multi-core machines to assist optimal performance. How best, then, to address these conditions?

You could rewrite your application to take advantage of parallel threads on multiple processors (while avoiding synchronization issues). Rewriting code is an option that you probably want to undertake only after some thoughtful planning. While you plan a strategy for your code, you can provide another primary component of your application that is *already* optimized for multi-core: the database.

As previously explained, you do not need to re-compile or re-architect your code to take advantage of Pervasive PSQL v11. The multi-core features of Pervasive PSQL v11 can help offset any performance degradation your end users might experience from your application not being optimized for multi-core environments. And that buys you some time while you plan your future development.

## Relational Interface Support for 64-bit Architecture

Pervasive PSQL v11 supports the relational interface (SRDE) on native 64-bit operating systems running on machines with 64-bit architecture. (Previous versions of Pervasive PSQL already included 64-bit support for the transactional interface through the Btrieve API and DTI.)

The transactional and relational interfaces for 64-bit Pervasive PSQL Server are now in the same common address space. In previous versions, the relational service interface was 32-bit only, which necessitated separate processes.

Pervasive PSQL now provides 32-bit and 64-bit versions of the Server Engine and the Client for the relational and the transactional interfaces. The Workgroup Engine is available only in a 32-bit version.



---

**Note** For the Beta Release, the relational support for 64-bit architecture is for Windows platforms only. The feature is not yet available for Linux.

---

### **Windows Registry**

Registry changes apply more to the operating system and less to how you interact with Pervasive PSQL. However, because Pervasive PSQL runs on 64-bit environments, some general information about the Windows registry is useful background.

On 64-bit operating systems the registry is split at certain important nodes into a 32-bit section and a 64-bit section. Access to registry keys is redirected by Windows to the appropriate section depending on whether the calling application is 32-bit or 64-bit. The Windows API allows applications to request which specific section to access. Refer to your operating system documentation for specifics about registry architecture on 64-bit platforms.

The Pervasive PSQL components transparently access the 32-bit or 64-bit section of the registry as required.

### **Pervasive PSQL Settings**

When Pervasive PSQL v11 is installed on a 64-bit Windows operating system, most components store their registry entries in the

64-bit section of the registry. In the 64-bit version of Pervasive PSQL, both 32-bit and 64-bit versions of certain components are present. Both versions read their settings from the same section of the registry. For example, if you enable debug tracing for the client components, the tracing applies to both 32-bit and 64-bit client components.

To read or modify commonly used settings, use Pervasive PSQL Control Center or the DTI API.

## ODBC Interface Support for 64-bit Architecture

Pervasive PSQL v11 supports the ODBC interface on native 64-bit operating systems running on machines with 64-bit architecture. The support for the 64-bit ODBC interface is for Windows platforms only. The feature is not yet available for Linux.

### **ODBC and Data Source Names (DSNs)**

On 64-bit Windows operating systems, 64-bit DSNs are distinct from 32-bit DSNs because of the registry design (see “Windows Registry”). Windows ODBC Data Manager requires that you know the bit architecture (called “bitness”) of your application and create a DSN with that same bitness. Pervasive PSQL v11 adopts this same model.

Pervasive PSQL provides dynamic link libraries (DLLs) used for setting up DSNs. The setup DLLs are used by ODBC Administrator and, for simplicity, are referred to as ODBC drivers. Pervasive PSQL v11 provides three ODBC drivers, as shown in the following table.

*Table 1-1 Pervasive PSQL ODBC Drivers for Windows*

ODBC Driver	PSQL Product Installed With	Behavior for All Products Installed With
32-bit Engine Interface	Server 64-bit Server 32-bit Workgroup	<ul style="list-style-type: none"> <li>Creates 32-bit engines DSNs</li> <li>Connects to a local named database</li> <li>Deprecated in Pervasive PSQL v11, as explained below</li> </ul>
32-bit Client Interface	Server 64-bit Server 32-bit Client 32-bit Workgroup	<ul style="list-style-type: none"> <li>Creates 32-bit Client DSNs</li> <li>Connects to a local or remote Engine DSN or to a named database</li> <li>ODBC Administrator still lists both Engine DSNs and named databases</li> </ul>
64-bit Client Interface	Server 64-bit Client 64-bit	<ul style="list-style-type: none"> <li>Creates 64-bit Client DSNs</li> <li>Connects to a local or remote named database</li> </ul>

Since the majority of SQL application interfaces (such as JDBC, ADO, and ADO.NET) connect to a named database, Pervasive PSQL v11 is moving toward this standard as follows:

- Deprecating 32-bit Engine DSNs. The 32-bit Engine Interface driver is still provided in this release, primarily for backwards compatibility. Pervasive recommends, rather than using Engine DSNs, that new or revised 32-bit applications connect to a named database through a Client DSN or use a DSN-less connection by specifying “Pervasive ODBC Client Interface.”
- Deprecating the DTI functions that manage 32-bit Engine DSNs. See “DTI.”
- Providing a 64-bit interface driver only for named databases. The 64-bit Client Interface can connect to a local named database, thus replacing the function of the Engine DSN, or to a remote named database. Connection to an Engine DSN is not supported.

### Frequently Asked Questions

The following table answers some frequently asked questions (FAQs) about the ODBC and DSN support in Pervasive PSQL v11.

*Table 1-2 FAQs About ODBC and DSN Changes*

Question	Answer
What happens to my existing 32-bit Engine DSNs when I upgrade to Pervasive PSQL v11 Server or Workgroup?	<p>No migration steps are required. Existing 32-bit Engine DSNs remain in place and continue to work as configured.</p> <p>Applications on the PSQL Server or Workgroup machine continue to work with 32-bit Engine DSNs.</p>
What happens to my existing 32-bit Client DSNs when I upgrade to Pervasive PSQL v11 Client?	<p>No migration steps are required. Existing Client DSNs continue to connect to remote Engine DSNs.</p> <p>If you edit a Client DSN with ODBC Administrator, you have the option to continue using a remote Engine DSN or to use a remote named database. See “ODBC DSN Setup GUIs.”</p> <p>Note, however, the recommendation is that new or revised 32-bit applications should connect to a named database, not to an Engine DSN. This conforms to industry norms.</p>
Are connections that use “Pervasive ODBC Client Interface” affected (so called “DSN-less” connections)?	<p>No. DSN-less connections that connect using “Pervasive ODBC Client Interface” continue to work.</p>

Table 1-2 FAQs About ODBC and DSN Changes *continued*

Question	Answer
<p>What about connections from PSQL Clients of previous releases (such as a PSQL v10.x Client)?</p>	<p>Pervasive PSQL v11 still supports remote Client DSNs, so clients from previous versions can still connect.</p> <p>Note, however, Engine DSNs are only 32-bit. 64-bit Engine DSNs cannot be created with Pervasive PSQL.</p>
<p>What do I need to do about DSNs if I port my 32-bit application to 64-bit?</p>	<p>No changes are required if the application uses DSN-less connections that connect using "Pervasive ODBC Client Interface."</p> <p>If the application uses DSNs, you must create 64-bit Client DSNs that connect to a named database.</p>
<p>What about the DSNs for the Demodata sample database installed with the database engine?</p>	<p>If you install Pervasive PSQL Server 64-bit, both 32-bit and 64-bit Client DSNs are created for the sample database.</p> <p>If you install Pervasive PSQL Client 64-bit on top of Pervasive PSQL Server 32-bit or on top of Pervasive PSQL Workgroup, no 64-bit DSNs are created. Only the DSNs created by the installation of the 32-bit database engine are present.</p> <p>Similarly, If you install Pervasive PSQL Server 32-bit or Pervasive PSQL Workgroup on top of Pervasive PSQL Client 64-bit, no 64-bit DSNs are created. Only the DSNs created by the installation of the 32-bit database engine are present.</p>
<p>Why do I not see my DSNs in ODBC Administrator?</p>	<p>On 64-bit Windows operating systems, 64-bit system DSNs are distinct from 32-bit system DSNs because of the registry design.</p> <p>If you are using the 64-bit ODBC Administrator, you will not see the 32-bit system DSNs, and vice versa.</p> <p>Note that, when the relational service interface on a 64-bit operating system receives a connection from a client to an Engine DSN, the database engine looks up the requested Engine DSN only in the 32-bit registry.</p> <p>See "Windows Registry" and "ODBC DSN Setup GUIs."</p>

Table 1-2 FAQs About ODBC and DSN Changes *continued*

Question	Answer
What if my application uses DTI to manage DSNs?	See “DTI”
What are the changes to ODBC Administrator?	See “ODBC DSN Setup GUIs”
Other than ODBC Administrator, does Pervasive PSQL v11 include new utilities to support 64-bit ODBC and DSNs?	Not for the Beta Release.
Are there any changes to existing utilities to support 64-bit ODBC and DSNs?	Yes. See “Utilities Affected.”
Do some descriptor fields that can be set through the various ODBC SQLSet and SQLGet functions accommodate 64-bit values while others are still 32-bit values?	<p>Yes, if you are using the 64-bit ODBC driver. Ensure that you use the appropriate sized variable when setting and retrieving descriptor fields. For more information, refer to the Microsoft ODBC documentation. See especially <a href="http://msdn.microsoft.com/en-us/library/ms716287%28VS.85%29.aspx">http://msdn.microsoft.com/en-us/library/ms716287%28VS.85%29.aspx</a>.</p> <p>A point of clarification is that SQL_ROWSET_SIZE is supported by both SQLGetStmtOption and SQLGetStmtAttr. If you are using the 64-bit ODBC driver and you call either SQLGetStmtOption or SQLGetStmtAttr, a 64-bit value is returned in *ValuePtr when that attribute parameter is set to SQL_ROWSET_SIZE.</p>
Going forward, is there a recommended strategy for ODBC connections?	<p>Yes. New or revised 32-bit applications, local or remote, should connect to a named database through a Client DSN, not to an Engine DSN. Alternately, applications could use DSN-less connections by specifying “Pervasive ODBC Client Interface.”</p> <p>This positions your application for the future when Engine DSNs will no longer be supported in Pervasive PSQL.</p>

## DTI

The DTI functions for DSNs manage only 32-bit Engine DSNs. Therefore, the following DTI functions are deprecated along with the 32-bit Engine Interface ODBC driver:

- “PvCreateDSN()”
- “PvCreateDSN2()”
- “PvGetDSN()”
- “PvGetDSNEx()”
- “PvGetDSNEx2()”
- “PvDeleteDSN()”
- “PvListDSNs()”
- “PvModifyDSN()”
- “PvModifyDSN2()”

All of these functions operate only on the 32-bit registry. This applies even if a 32-bit database engine is installed on a 64-bit operating system. The 32-bit ODBC Administrator uses these DTI functions. Therefore, the list of existing Engine DSNs and newly created Engine DSNs are only for the 32-bit registry.

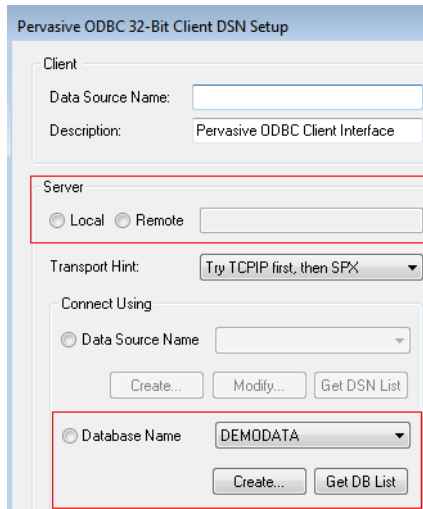
See *Distributed Tuning Interface Guide* for an explanation of the functions that manage DSNs.

## ODBC DSN Setup GUIs

The following changes apply to setting up DSNs through ODBC Administrator.

- A new DSN setup graphical user interface (GUI) is available for setting up 64-bit Client DSNs. See also Table 1-1, “Pervasive PSQL ODBC Drivers for Windows.”

- The GUI for setting up 32-bit Client DSNs has been modified to allow selection of database name for a local or remote server. See also Table 1-1, “Pervasive PSQL ODBC Drivers for Windows.”



- All of the GUIs for setting up DSNs now list in the title bar the DSN bitness to which the GUI applies:
  - Pervasive ODBC 64-bit Client DSN Setup
  - Pervasive ODBC 32-bit Client DSN Setup
  - Pervasive ODBC 32-bit Engine DSN Setup

See the chapter “DSNs and ODBC Administrator” in *SQL Engine Reference* for a discussion of the new controls on the GUIs.

## ODBC Header Files

The `sql.h` and `sqltypes.h` header files for ODBC contain differences for the compilation of 32-bit and 64-bit applications. Refer to the ODBC documentation on the Microsoft Web site for a discussion of 64-bit ODBC. For example, you may find the following information useful: [http://msdn.microsoft.com/en-us/library/ms716287\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/ms716287(VS.85).aspx).

## **Utilities Affected**

For Pervasive PSQL Server and Client installations on 64-bit operating systems, Pervasive PSQL Control Center (PCC) contains separate choices for 32-bit and 64-bit ODBC Administrator. The choices are available on the Tools menu. See “Additional Utilities” in *Pervasive PSQL User's Guide*.

In addition, the option to create a DSN on the New Database dialog is now qualified for 32-bit: “Create 32-bit Engine DSN.” See “New Database GUI Reference” in *Pervasive PSQL User's Guide*. (PCC is a 32-bit application. A 64-bit version of it is not available.)

The Pervasive ODBC DSN setup GUIs have changed. See “ODBC DSN Setup GUIs.”

## Product Activation

Product activation is a validation process verifying that the copy of the software is legitimate, correctly licensed and on the appropriate hardware and software platform. Pervasive PSQL v11 includes the following additions to product activation:

- “Telephone Activation”
- “Product Activation for OEMs”

### **Telephone Activation**

If Pervasive PSQL Server or Workgroup is installed on a system that has no Internet connectivity, directly or indirectly, the product can be activated by telephone with the assistance of Technical Support. The toll free number at Pervasive is 800 287-4383.

Telephone activation is available during regular United States office hours, Central Standard Time. Calls received during off-hours or holidays are returned the next business day.

See “Telephone Activation” in *Pervasive PSQL User's Guide*.



---

**Note** For the Beta Release, telephone activation is available only on Windows platforms.

---

### **Product Activation for OEMs**

Pervasive PSQL v11 extends the product activation technology to our original equipment manufacturer (OEM) partners. If you are an OEM partner, refer to the following resources:

- Product activation information on the Pervasive Web site: <http://www.pervasivedb.com/Database/support/Pages/Activation.aspx>
- OEM Web Portal on the Pervasive Web site. The Portal allows you to generate product keys and perform various administrative functions pertaining to keys. The Portal is available 24/7 and provides an easy-to-use interface. See also on the Portal:
  - *OEM Partner Handbook*, which has been extensively revised.
  - *Product Activation for OEM Partners* white paper.
  - *Product Activation Troubleshooting Guide for OEM Support Staff*.

## Configuration Settings

The range and default for the Communications Threads setting have changed:

- The range is now *num\_cores* to 256, where *num\_cores* is the number of processors in the machine on which the database engine is running.
- The default is *num\_cores*.

Previously, the range was 1 to 1,024 and the default was 16. See also “Communications Threads.”

## Utility Changes

Pervasive PSQL v11 includes changes to the following utilities:

- “Pervasive PSQL Control Center”
- “ODBC Administrator”

### ***Pervasive PSQL Control Center***

Pervasive PSQL Control Center (PCC) contains the following change pertaining to DSNs.

- On Pervasive PSQL Server 64-bit installations, the PCC Tools menu contains separate choices for 32-bit and 64-bit ODBC Administrator.
- The option to create a DSN on the New Database dialog is now qualified for 32-bit: “Create 32-bit Engine DSN.”

See also “ODBC and Data Source Names (DSNs).”

### ***ODBC Administrator***

The Pervasive ODBC setup GUIs for 32-bit DSNs have changed. A new ODBC setup GUI for 64-bit DSNs is available. See “ODBC DSN Setup GUIs.”

## Deprecated and Discontinued Features

### ***Deprecated Features***

The following categories discuss features that are deprecated in Pervasive PSQL v11. Although the features are still available in Pervasive PSQL v11, they will be removed from the product in a future release. Plan accordingly for new application development and revisions to existing applications.

#### **ODBC**

The following ODBC features are still available in Pervasive PSQL v11 but will be removed from the product in a future release.

- 32-bit Engine DSNs. See “ODBC and Data Source Names (DSNs).”
- DTI functions that manage 32-bit Engine DSNs. See “DTI.”

#### **Pervasive Direct Access Components (PDAC)**

The following dynamic PDAC libraries are still available in Pervasive PSQL v11 but will be removed from the product in the future.

- PDAC dynamic libraries for Delphi 2006 and 2007
- PDAC dynamic libraries for C++ Builder 6

### ***Discontinued Features***

The following features are no longer supported in Pervasive PSQL v11.

- Support for Windows 2000



# *Index*

## **A**

- Activation
  - by telephone 1-16
- Architecture
  - 64-bit ODBC 1-9
  - 64-bit relational support 1-7

## **C**

- Communications Thread
  - changes for multi-core support 1-3

## **D**

- Deprecated features 1-19
- Discontinued Features 1-19
- Driver
  - ODBC 1-9
- DSN
  - deprecated DTI functions 1-12
  - DSN-less connections 1-10
  - Engine DSNs and deprecated DTI functions 1-12
  - ODBC Administrator differences 1-11
  - porting 32-bit application to 64-bit 1-11
  - setup GUIs 1-13
- DTI
  - deprecated functions 1-12

## **K**

- Key
  - product activation and 1-16

## **M**

- Multi-core
  - configuration settings affected by 1-3
  - false sharing issues 1-5
  - memory contention issues 1-4
  - multiple threads issues 1-5
  - performance and parallel threads 1-2
  - performance and scalability 1-3
  - primary component affecting 1-2
  - support for 1-2

## **O**

- ODBC
  - Administrator and DSNs 1-11
  - and 64-bit architecture 1-9
  - deprecated features in Pervasive PSQL 1-9
  - DSN setup GUIs 1-13
  - DSN-less connections 1-10
  - frequently asked questions 1-10
  - migration of existing DSNs 1-10
  - Pervasive PSQL drivers for 1-9

## **P**

- Product Activation
  - technology extended to OEMS 1-16
- Product activation 1-16

## **R**

- Registry
  - nodes for 32-bit and 64-bit 1-7
- Relational interface
  - support for 64-bit architecture 1-7

## **T**

- Telephone activation 1-16

## **U**

- Utility
  - changes for Pervasive PSQL v11 1-18

