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: database@pervasive.com
Web: http://www.pervasivedb.com

PERVASIVE®

What's New in Pervasive PSQL
General Release September 2010
138-004435-001

Contents

Tables

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 PSQL Web site, www.pervasivedb.com.

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

chapter 1

What Is New in Pervasive PSQL v11

An Overview of New and Changed Features

The General Release includes the following new features and changes:

- Multi-core Support
- Support for IPv6
- 64-bit ODBC Driver
- Support for .NET Framework 3.5 SP1 and 4.0
- PDAC Development Environments
- Enhancements to Other SDK Access Methods
- Product Authorization
- Configuration Settings
- Utility Changes
- Deprecated and Discontinued Features

Multi-core Support

Pervasive PSQL v11 is specifically designed to increase scalability and performance on multi-core machines. Install Pervasive PSQL v11 on a multi-core machine and the benefits are immediately available in a multiuser environment.

You may wonder “what benefits?” Increased scalability and performance are obviously desirable and assumed to be available with advances in hardware technology. Heretofore, advances in hardware technology meant advances in speed. Applications just ran faster. Today, advances in computing technology mean increased parallelism and not increased clock speeds. And that presents challenges to which your application has probably never had to contend.

The rules have not just changed because of multi-core environments, they have changed dramatically. For example, applications that share data with multiple users and use a database where transactional integrity must be maintained can run *slower* on multi-core processors.

Because the majority of applications using Pervasive PSQL fall into that category, multi-core support is a primary feature of Pervasive PSQL v11. It is of primary importance to you as you transition your multiuser applications into multi-core environments.

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 multiuser 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 the hardware has a significant impact if it fundamentally differs from its predecessors.

But tuned in the right way, applications that would otherwise be slowed down can take advantage of hardware changes and experience significant performance improvement. In many cases, swapping out portions of the application stack, such as the database, can address multi-core issues with no immediate changes required to the application. This approach provides as a low risk way to buy time while you plan longer-term strategies for application development.

Using Pervasive PSQL v11 as the database, you can realize increased performance and scalability on multi-core machines.

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 recompile or rearchitect 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 through architecture designs made specifically for multi-core hardware. 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 recompile or rearchitect your code.

Configuration Settings

The majority of multi-core improvements in Pervasive PSQL v11 are transparent. You are not required to adjust any settings to further enhance the optimizations. The configuration setting "Communications Threads" has changed and can be used to fine-tune performance if you choose. See Configuration Settings.

The Multi-core Dilemma

Several common problems are at play in the multi-core world of hardware and software interaction that may cause *decreased* performance with your application. Among them are multiple threads and memory contention. For a thorough discussion of these and other problems, refer to the white paper *The Multi-core Dilemma* by Dan Woods, CTO of CITO Research. The white paper is available on the Pervasive Web site.

A brief discussion in this document of multiple threads and memory contention illustrates why multi-core support is a primary feature of Pervasive PSQL v11.

Multiple Threads

A multithreaded application does not necessarily run better on a multi-core machine. In fact, you may find that your multithreaded application runs slower.

To work correctly 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 efficiency. 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. Caching does not improve the problem; it makes it worse (see Memory Contention).

Where possible, each core should process separate data. Otherwise, the overhead associated with synchronization can slow down performance significantly. Recall that Pervasive PSQL v11 has been architected to provide parallel threads that are synchronized.

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.

If multiple cores or processors have caches that point to the same data and one core modifies the data, the cached data on the other core is no longer valid, and the caches must be synchronized. Contention also occurs as the processors repeatedly check the caches 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.

Recall that with Pervasive PSQL v11 activities of multiple users proceed on independent server CPUs as a way to reduce memory contention. Multiple users can read the same cached file pages simultaneously and access independent files.

The Role of the Operating System

You may be wondering how much the operating system (OS) assists with the problems of multiple cores. Less than you would guess, even with current 64-bit ones.

When contention for resources happens, the OS handles the resolution. For the majority of applications, the OS handles thread contention slower on multi-core systems. That is, the OS on multi-core systems take a longer time to resolve the contention points.

Why is this? An OS optimized for multi-core does not fix your problems if your applications still require the operating system to perform tasks in a single-file fashion.

When the OS gets requests from an application that do not incorporate instructions for multi-core processing, the OS is very cumbersome at sorting out the sequence in which the requests are processed. This is analogous to a traffic jam on a highway. Conceptually, the OS asks *each* waiting driver whether or not they are ready to go before allowing the vehicle to proceed. Although such processing jams are occurring at the OS level, users perceive the slowdown as an application performance problem.

An application optimized for multi-core provides instructions for the OS on how to manage shared resources and determine priority for access to those resources. Information requests are organized in such a way that they do not compete for cache lines or access to central memory.

Recall that Pervasive PSQL v11 includes architecture designs made specifically for multi-core hardware. Low-level locking has been optimized for multi-core machines.

Benefiting from the Present While Planning For the Future

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?

Ultimately, applications will have to be rearchitected to perform optimally on multi-core machines. This allows the application to take advantage of parallel threads on multiple processors while avoiding synchronization issues.

Rearchitecting takes thoughtful planning and time to implement, perhaps even years. Meanwhile, business continues. As mentioned at the beginning of this section, multi-core support becomes of primary importance to you as you transition your applications into multi-core environments.

Your “application” consists of your code, the database, the operating system, and hardware. Hardware systems have already addressed multi-core support. Operating system provide some assistance provided your application takes advantage of the multiple cores. That leaves the database.

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. In most cases, you can boost application performance without having to recompile or change your application code.

Support for IPv6

Internet Protocol version 6 (IPv6) is the next-generation Internet Protocol version designated as the successor to IPv4. This section discusses the following topics:

- Using Pervasive PSQL With IPv6
- Frequently Asked Questions About IPv6 Support
- Pervasive PSQL Utilities and IPv6
- IPv6 Aspects for Application Programmers

Using Pervasive PSQL With IPv6

Pervasive PSQL v11 supports IPv6 for the following access methods on Windows operating systems:

- Transactional (also known as Btrieve)
- DTI (Distributed Tuning Interface)

Both access methods function correctly in an IPv4 environment, an IPv6 environment, or an environment that combines the two. No special configurations of Pervasive PSQL are required.

Client Connections

A Pervasive PSQL Client connects to a IPv6 host running the Pervasive PSQL database engine the same way as for IPv4. That is, the Client specifies a server and connects through DTI or by specifying a URI or UNC. The server can be either the name or IP address of the machine running Pervasive PSQL Server or Workgroup.

See also the following:

- Database URIs in *Pervasive PSQL Programmer's Guide*.
- Universal Naming Convention (UNC) Path Formats in *Getting Started With Pervasive PSQL*.
- Making a Connection to a Server Using DTI in *Distributed Tuning Interface Guide*.

The following topics in this subsection discuss how to specify a server using IPv6 addresses.

IPv6 Address Formats

Raw IPv6 addresses can be written as 8 colon-separated segments where each segment is a a 4-digit hexadecimal value. For example, 1234:5678:90ab:cdef:1234:5678:90ab:cdef.

Pervasive PSQL supports only unicast addresses. The following are the unicast address formats that can be used with Pervasive PSQL.

Table 1-1 IPv6 Unicast Address Formats Supported by Pervasive PSQL

Unicast Address Format	Description
Loopback	The local loopback address, which in IPv6 is 0:0:0:0:0:0:0:1. The loopback address can be abbreviated to ::1. The IPv6 loopback address is equivalent to the IPv4 loopback address of 127.0.0.1.
Global	Global addresses have a 64-bit prefix where the first 3 bits are always 001, the next 45 bits are set to the global routing prefix, the next 16 bits are set to the subnet ID and the last 64-bits are the interface ID. Example: 2001:db8:28:3:f98a:5b31:67b7:67ef
Link Local	Link Local addresses are used by nodes when communicating with neighboring nodes on the same link. Link Local addresses have a 64-bit prefix where the first 10 bits are set to 1111 1110 10, the next 54 bits are set to 0 and the last 64 bits are the interface ID. The link local prefix is often represented as FE80::/64. Example: fe80:0:0:0:713e:a426:d167:37ab (which may also be specified as fe80::713e:a426:d167:37ab) See also Restrictions.

IPv6 Address Modifiers

IPv6 includes address modifiers which can act as shortcuts, or to specify the destination in more detail. Pervasive PSQL supports the following ones for IPv6.

Modifier	Explanation
::	Represents one or more colon-separated zeroes. For example, ::1 is equivalent to 0:0:0:0:0:0:0:1. The :: modifier can be used only once within an IPv6 address.
%	Represents the ZoneID or interface of a destination node. A ZoneID is an integer that specifies the zone of the destination for IPv6 traffic. See Restrictions.

IPv6 With UNC Paths and URI Connections

UNC paths do not allow certain special characters, such as colons. Since raw IPv6 addresses use colons, different methods of handling UNC paths are available. Pervasive PSQL supports the following methods:

- IPv6-literal.net Names
- Bracketed IPv6 Addresses

IPv6-literal.net Names

An ipv6-literal.net name is a raw IPv6 address with three changes:

- ":" is replaced with "-"
- "%" is replaced with "s"
- The whole address is appended with ".ipv6-literal.net"

Examples:

Initial Addresses	fe80::713e:a426:d167:37ab%4
	2001:db8:28:3:f98a:5b31:67b7:67ef
Modified Addresses	fe80--713e-a426-d167-37abs4.ipv6-literal.net
	2001-db8-28-3-f98a-5b31-67b7-67ef.ipv6-literal.net

Ipv6-literal.net names are allowed in a URI or UNC used with Pervasive PSQL.

Bracketed IPv6 Addresses

A bracketed IPv6 address is a raw IPv6 address with square brackets around it. This format is also referred to as a UNC-safe address.

Examples:

Initial Addresses	fe80::713e:a426:d167:37ab%4
	2001:db8:28:3:f98a:5b31:67b7:67ef
Modified Addresses	[fe80::713e:a426:d167:37ab%4]
	[2001:db8:28:3:f98a:5b31:67b7:67ef]

The use of square brackets is required for raw IPv6 addresses used in a URI or UNC with Pervasive PSQL. See Restrictions. Note that if you use an address with a ZoneID in a URI, the ZoneID character "%" must use the escape characters "%25." See Restrictions.

Restrictions

The following table lists the restrictions on the use of IPv6 with Pervasive PSQL.

Table 1-2 IPv6 Restrictions With Pervasive PSQL

Restriction	Discussion
The Pervasive PSQL Server Engine in an IPv6-only environment	The Pervasive PSQL Server Engine is not supported in an IPv6-only environment on Windows Server 2003 or Windows XP operating systems. The Server Engine may be used with IPv6 on Windows Vista, Windows 7, and Windows Server 2008. With Windows Server 2003 or Windows XP, status code 170 is returned when a PSQL Client attempts to connect to the PSQL Server. NetBIOS over TCP (NetBT) is not implemented for IPv6 on Windows Server 2003 or Windows XP. If you want to use PSQL Server on Windows Server 2003 or Windows XP, the network environment must be IPv4 or a combination of IPv4 and IPv6. This restriction applies only to the Server Engine. Pervasive PSQL Workgroup may be used with IPv6 on Windows Server 2003, Windows XP, Windows Vista, Windows 7, and Windows Server 2008.
Square brackets are required for raw IPv6 addresses when the address is used in a URI or UNC	Raw IPv6 addresses, abbreviated or not, must be enclosed by square brackets if the address is used in a URI or UNC. Examples: • btrv://czjones@[2001:b1::23]/demodata • btrv://abanderas@[2001:12:34:56:78:90:12:23]/demodata • \\[2001:12:34:56:78:90:12:23]\acctsvr1\Domestic\file.mkd Failure to bracket the IPv6 address results in status code 3014 or 3103 for Btrieve calls using a URI, or status code 11, 94, or 170 for Btrieve calls using a UNC.
In a URI, if you include a ZoneID to a server address, the “%” ZoneID character must be escaped with “%25”	If you use a btrv:// connection with an IPv6 address, you must escape the ZoneID for the host name. Example: A UNC-safe addresses like btrv://@[fe80::20c:29ff:fe67:2ee4%4] must be changed to btrv://@[fe80::20c:29ff:fe67:2ee4**%25**4]

Table 1-2 IPv6 Restrictions With Pervasive PSQL

Restriction	Discussion
PCC usage in an IPv6-only environment	In an IPv6-only environment, PCC allows only the functionality supported by the transactional or DTI access methods. For example, you can connect a PSQL Client from an IPv6-only machine to a database engine on an IPv6-only server machine. PCC allows you to view and set Engine and Client properties because those features use DTI. However, you cannot browse databases or use Table Designer because those feature use other access methods, such as the relational interface, which are not yet supported for IPv6.
License Administrator (and clilcadm)	The Pervasive licensing server does not yet support IPv6. Because of this, you can use License Administrator over IPv6 to administer licenses but you cannot authorize a license with the utility. To authorize a license, you must use an IPv4 network, remote authorization, or telephone authorization.

Frequently Asked Questions About IPv6 Support

The following table answers some frequently asked questions (FAQs) about IPv6 support in Pervasive PSQL v11.

Table 1-3 FAQs About IPv6 Support

Question	Answer
Can I use Pervasive Auto Reconnect (PARC) with IPv6?	Yes.
Does Pervasive PSQL support IPv6 communications in virtual machine environments?	Yes.
Does IPv6 support apply to the relational access method (SRDE)?	No. Only the transactional and DTI access methods are supported.
Is IPv6 supported for Linux distributions or Macintosh OS X?	No. Only Windows platforms are supported.
Is IPv6 supported for Pervasive DataExchange, AuditMaster, and Backup Agent?	No.

Table 1-3 FAQs About IPv6 Support continued

Question	Answer
Does a network environment that includes both IPv4 and IPv6 affect Pervasive PSQL user counts?	No. Pervasive PSQL Server or Workgroup uses one user count for each *unique* incoming protocol from the same client computer session (such as TCP/IP and SPX). IPv4 and IPv6 are just different address formats of TCP/IP.
Can the Listen IP Address configuration setting be set to multiple addresses?	Yes. See Listen IP Address.

Pervasive PSQL Utilities and IPv6

The following Pervasive PSQL utilities support IPv6. No special configuration of them is required.

Utility	See Also
bcfg	Configuration Reference in *Advanced Operations Guide*
Function Executor	Testing Btrieve Operations in *Advanced Operations Guide*
License Administrator (and clilcadm)	License Administration in *Pervasive PSQL User's Guide* See Restrictions.
Monitor (and bmon)	Monitoring Database Resources in *Advanced Operations Guide*
Pervasive PSQL Control Center (PCC)	Using Pervasive PSQL Control Center in *Pervasive PSQL User's Guide* If you are using PCC in an IPv6-only environment, see Restrictions.

IPv6 Aspects for Application Programmers

Because IPv6 has not been widely adopted, the section discusses a few aspects of it that an application programmer may want to investigate further. The intent is not to explain in detail networking concepts or IPv6, but to provide a very brief introduction to IPv6. For a complete discussion of IPv6, see the IPv6 specification at www.ipv6.org, and refer to the IPv6 documentation from the various operating system vendors and network hardware vendors.

Importance of IPv6

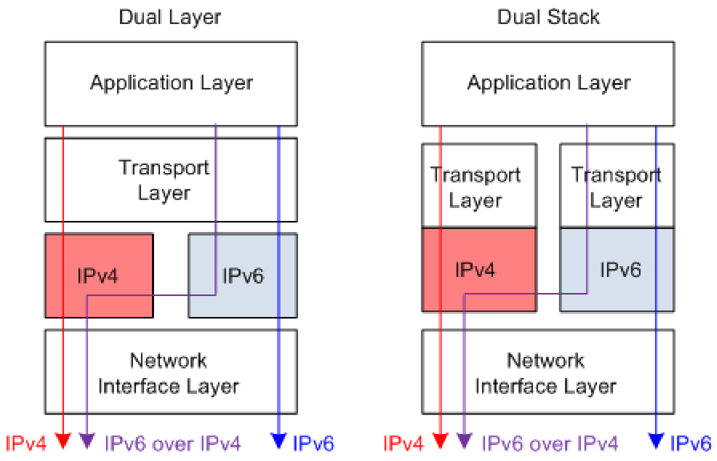
IPv6 is the next-generation Internet Protocol version designated as the successor to IPv4. IPv4 was the first implementation used in the Internet that is still in dominant use currently. Because of the age of IPv4, and the changing world environment of networking, IPv4 contains several limitations that make it unsuitable for future needs.

Perhaps the most serious limitation is that its address space will eventually be exhausted. Even today, public IPv4 addresses have become relatively scarce. In addition, world-wide networking has introduced requirements beyond what IPv4 provides, such as simpler configuration capabilities, enhanced security, and extensibility.

IPv6 addresses the shortcomings of IPv4 as well as offering a host of additional benefits. Newer hardware and operating systems provide IPv6 support. Applications for certain sectors already require IPv6 support. For example, the governments of the United States and Japan have mandated support for IPv6. Since IPv4 must eventually be replaced, the sooner that occurs, the sooner the benefits of IPv6 can be realized.

Client/Server Communications

During the transition period between IPv4 and IPv6 support, both protocols will likely be functional on certain operating systems. Depending on the operating systems, this is referred to as dual layer or dual stack. Note, however, that IPv4 and IPv6 traffic is independently routed. For two hosts to communicate, both must either be capable of using IPv4 or capable of using IPv6.



Dual Layer	Dual Stack
• Available on Windows Vista, Windows Server 2008, and Windows 7 • IPv6 automatically installed with the operating system • IPv6 cannot be uninstalled • IPv6 can be turned off • IPv4 can be turned off	• Available on Windows Server 2003 and Windows XP (and Linux distributions) • IPv6 must be installed as an add-on for Windows platforms • IPv6 can be uninstalled on Windows platforms • IPv6 can be turned off • IPv4 cannot be turned off

If you want to configure the network settings at the operating system level, note the following.

Operating System	IPv6 Notes
Windows Server 2003 and Windows XP	IPv6 must be manually installed. No network GUI utilities are available, but the following command-line utilities are provided: ipconfig, netsh and nsupdate.
Windows Vista and newer	Network GUI configuration utilities available, as well as the command-line utilities ipconfig, netsh and nsupdate.

Host Files, ZoneIDs, and Name Discovery

Within the hosts file, each IP uses only lines with compatible address formats. For example, if you request an IPv4 address for a host name, the IPv6 lines are ignored. Compatible addresses also apply for localhost, so a hosts files typically has localhost lines for 127.0.0.1 (IPv4) and ::1 (IPv6).

When doing a lookup to convert a name into an address, the application programmer specifies whether to use IPv4, IPV6 or both. A networking component of the operating system uses administrator-level preferences to determine how to sequence the lookups to the local hosts file, the local DNS cache, the remote DNS server, and so forth. With IPv6, there are new auto-discovery protocols that can find remote machines without using DNS.

You can specify an IPv6 address in a hosts file with the following restrictions:

- Records in a hosts file cannot include the ZoneID
- The hosts file can have separate lines for IPv4 and IPv6 with the same node name.

The use of hosts files is most useful when ZoneIDs are not required.

ZoneID

The ZoneID maps to a network interface. With a single network interface card (NIC) and gateway, a ZoneID is not needed because the gateway is reached by only one route. Most machines enabled for IPv6 have multiple interfaces because of built-in support for transition routers like ISATAP, 6to4, or Teredo.

In netsh commands, you must use the interface name (using the interface= parameter), for example “Local Area Connection 2” or “eth0.” When using ping with an IPv6 address, you may need to use the ZoneID, for example fe80::abcd%10, in which case the decimal integer 10 is the ZoneID.

On Windows platforms, you can display the ZoneIDs for each interface with the ipconfig command..

Name Discovery

IPv6 contains auto-discovery protocols that can find remote machines without using Domain Name System (DNS). The Link Local Multicast Name Resolution (LLMNR) is a protocol based on the DNS packet format. LLMNR allows both IPv4 and IPv6 hosts to perform name resolution for hosts on a single subnet without a DNS server. Since every IPv6 machine has a link-local address, LLMNR locates the machine on the subnet, if present, before having to perform a DNS lookup for a link-global address.

64-bit ODBC Driver

Pervasive PSQL v11 now supports the ODBC interface for 64-bit applications. The 64-bit ODBC driver is installed with Pervasive PSQL Server 64-bit and Pervasive PSQL Client 64-bit.

ODBC and Data Source Names (DSNs)

On 64-bit Windows operating systems, 64-bit DSNs are distinct from 32-bit DSNs because of the Windows registry design. 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. Therefore, 64-bit applications use the 64-bit ODBC driver and 32-bit applications use the 32-bit ODBC driver.

The application bitness does not have to match the bitness of the Pervasive PSQL Server product. For example, the 64-bit ODBC driver or the 32-bit ODBC driver can be used with either Pervasive PSQL Server 64-bit or Pervasive PSQL Server 32-bit.

Pervasive PSQL v11 provides three ODBC drivers, as shown in the following table.

Table 1-4 Pervasive PSQL ODBC Drivers for Windows

ODBC Driver	PSQL Product Installed With	Behavior for All Products Installed With
Pervasive ODBC Engine Interface	Server 64-bit Server 32-bit Workgroup	• Setup creates 32-bit Engine DSNs • Connects to a local named database • For use by 32-bit applications • Deprecated in Pervasive PSQL v11, as explained below
Pervasive ODBC Client Interface	Server 64-bit Server 32-bit Client 32-bit Workgroup	• Setup creates 32-bit Client DSNs • Connects to a local or remote named database or an Engine DSN • GUI lists both named databases and Engine DSNs • For use by 32-bit applications
Pervasive ODBC Interface	Server 64-bit Client 64-bit	• Setup creates 64-bit DSNs • Connects to a local or remote named database • For use by 64-bit applications

To simplify the method for connecting to a named database, Pervasive PSQL v11 includes the following enhancements:

- 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 ODBC 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 1-5 FAQs About ODBC and DSN Changes

Question	Answer
Is the 64-bit ODBC driver supported for Linux distributions or Macintosh OS X?	No. Only Windows platforms are supported as discussed in Table 1-4.
What happens to my existing 32-bit Engine DSNs when I upgrade to Pervasive PSQL v11 Server or Workgroup?	No migration steps are required. Existing 32-bit Engine DSNs remain in place and continue to work as configured. Applications on the PSQL Server or Workgroup machine continue to work with 32-bit Engine DSNs.
What happens to my existing 32-bit Client DSNs when I upgrade to Pervasive PSQL v11 Client?	No migration steps are required. Existing Client DSNs continue to connect to remote Engine DSNs. 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. Note, however, the recommendation is that new or revised 32-bit applications should connect to a named database, not to an Engine DSN since Engine DSNs are deprecated.
Are connections that use “Pervasive ODBC Client Interface” affected (so called “DSN-less” connections)?	No. DSN-less connections that connect using “Pervasive ODBC Client Interface” continue to work.

Table 1-5 FAQs About ODBC and DSN Changes continued

Question	Answer
What about connections from PSQL Clients of previous releases (such as a PSQL v10.x Client)?	Pervasive PSQL v11 still supports remote Client DSNs, so clients from previous versions can still connect. Note, however, Engine DSNs are only 32-bit for both Pervasive PSQL Server 32-bit and 64-bit. 64-bit Engine DSNs cannot be created with Pervasive PSQL.
What are the ODBC connection strings for Pervasive PSQL DSNs?	See ODBC Connection Strings in *SQL Engine Reference*.
What do I need to do about DSNs if I port my 32-bit application to 64-bit?	If the application uses DSN-less connections that connect using "Pervasive ODBC Client Interface," change the connection string to "Pervasive ODBC Interface." See ODBC Connection Strings in *SQL Engine Reference*. If the application uses DSNs, you must create 64-bit DSNs that connect to a named database.
What about the DSNs for the Demodata sample database installed with the database engine?	The installation of Pervasive PSQL Server 32-bit or Pervasive PSQL Workgroup creates a Client DSN for Demodata instead of an Engine DSN. The installation of Pervasive PSQL Server 64-bit creates both a 32-bit Client DSN and a 64-bit DSN for Demodata. 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. 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.
How do I run the 32-bit ODBC Administrator on a 64-bit operating system?	See ODBC Administrator in *SQL Engine Reference*.
Why do I not see my DSNs in ODBC Administrator?	On 64-bit Windows operating systems, 64-bit system DSNs are distinct from 32-bit system DSNs because of the registry design. If you are using the 64-bit ODBC Administrator, you will not see the 32-bit system DSNs, and vice versa. 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. See ODBC DSN Setup GUIs.
What if my application uses DTI to manage DSNs?	See DTI.

Table 1-5 FAQs About ODBC and DSN Changes continued

Question	Answer
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?	No.
Are there any changes to existing utilities to support 64-bit ODBC and DSNs?	Yes. See Utilities Affected by ODBC Changes.
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?	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 http://msdn.microsoft.com/en-us/library/ms716287%28VS.85%29.aspx. 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.
Going forward, is there a recommended strategy for ODBC connections?	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.” This positions your application for the future when Engine DSNs will no longer be supported in Pervasive PSQL.

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()

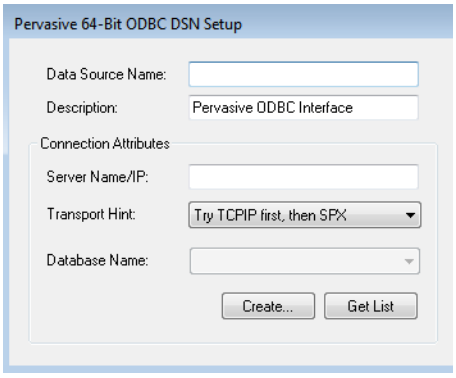
All of these functions operate only on the 32-bit registry. This applies even if a 64-bit database engine is installed on a 64-bit operating system. The 32-bit ODBC Administrator uses the DTI functions for Engine DSNs. Therefore, the list of existing Engine DSNs and newly created Engine DSNs are only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

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

ODBC DSN Setup GUIs

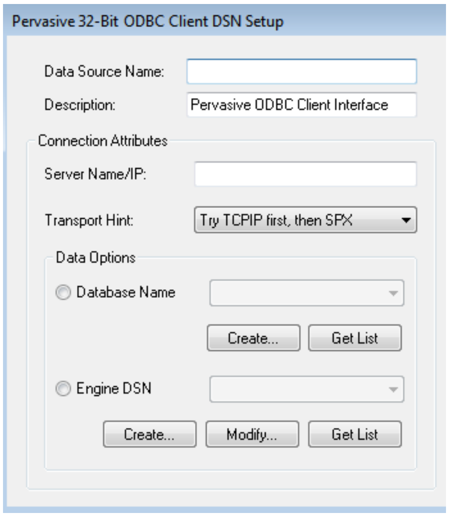
The following changes apply to setting up DSNs through ODBC Administrator.

- A new graphical user interface (GUI) is available for setting up 64-bit DSNs. See also Table 1-4, Pervasive PSQL ODBC Drivers for Windows.


Pervasive 64-Bit ODBC DSN Setup

Data Source Name:
Description: Pervasive ODBC Interface
Connection Attributes
Server Name/IP:
Transport Hint: Try TCPIP first, then SPX
Database Name:

- The GUI for setting up 32-bit Client DSNs has been modified as follows:
  - The GUI now allows selection of a local or remote server name or IP address. See also Table 1-4, Pervasive PSQL ODBC Drivers for Windows.



Description: Pervasive ODBC Client Interface

Transport Hint: Try TCPIP first, then SPX

  - The “Server” group box is now labeled “Connection Attributes”
  - The control labeled “Address” is now labeled “Server Name/ IP.”
  - The control labeled “Data Source Name” is now labeled “Engine DSN.”
  - The “Options” button is now labeled “Advanced” and displays the advanced connection attributes. The advanced connection attributes provide the same choices as were previously available on the Options dialog.

- The GUI for setting up Engine DSNs has been modified as follows:
  - The “Database” group box is now labeled “Connection Attributes”
  - The “Options” button is now labeled “Advanced” and displays the advanced connection attributes. The advanced connection attributes provide the same choices as were previously available on the Options dialog.

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.

Utilities Affected by ODBC Changes

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.

Support for .NET Framework 3.5 SP1 and 4.0

Pervasive PSQL v11 provides two versions of the ADO.NET Data Provider, version 3.2 and 3.5. Both versions are installed by default with the database engine and with the Pervasive PSQL Client.

The installation puts both Data Providers under the “Program Files (x86)” directory whether you are installing the 32-bit or 64-bit version of Pervasive PSQL. The Data Providers are bitness independent, however. Each Data Provider works with both the 32-bit and the 64-bit .NET Framework.

Pervasive PSQL ADO.NET Data Provider 3.2

The Pervasive PSQL ADO.NET Data Provider 3.2 has no new features from prior versions of the 3.2 Provider. It is included for application developers who want to use that version with Pervasive PSQL v11.

Pervasive PSQL ADO.NET Data Provider 3.5

Pervasive PSQL ADO.NET Data Provider 3.5 supports the new features in .NET Framework 3.5 SP1. The Provider is compliant with the .NET Framework Versions 2.0, 3.0, 3.5, 3.5 SP1, and 4.0. The 3.5 Provider does not support new features introduced in .NET Framework 4.0 but will run under the .NET Framework 4.0 with support for all the Entity Framework 1.0 features.

In addition, the Pervasive PSQL ADO.NET Data Provider 3.5 includes the following main features:

- Development using a set of methods tailored for the new Entity Framework consumers such as LINQ, EntitySQL, and ObjectServices.
- Pervasive Bulk Load. The DbBulkCopy class supports bulk loading data in the Common Programming Model. In addition, the Data Provider has a provider-specific bulk load class.
- Connection statistics support.
- A Schema Options connection string option to specify additional schema metadata to be returned.
- Native parameter marker and parameter binding support.

- Microsoft Enterprise Library 4.1 (October 2008) support, including data access application block (DAAB) support.
- Initial Command Timeout connection string option to specify an initial command timeout when a connection is established.
- Support for Microsoft Visual Studio 2008 and Visual Studio 2010.

See *Pervasive PSQL Data Provider for .NET Guide* in the SDK documentation for complete details.

PDAC Development Environments

Pervasive PSQL v11 includes PDAC for these additional development environments:

- RAD Studio 2009
- RAD Studio 2010

Support for RAD Studio 2009 and 2010 refers only to the Delphi and C++ Builder components supported by Pervasive PSQL v11.

Pervasive PSQL v11 no longer provides PDAC integration for Delphi and C++ Builder development environments version 6 and older. See Deprecated and Discontinued Features.

See also *Pervasive Direct Access Components Guide* in the SDK documentation.

Enhancements to Other SDK Access Methods

Pervasive PSQL v11 includes enhancements to the SDK access method Distributed Tuning Objects (DTO).

DTO

Pervasive PSQL v11 includes the following new methods.

DTO Object	Method	Description
DtoDatabase	AddUserToGroup	Adds an existing user to an existing group in the database
	AlterUserName	Changes an existing user's name in the specified database
	AlterUserPassword	Changes an existing user's password in the specified database
	CreateGroup	Creates a new user group in the existing database
	CreateUser	Creates a new user in the existing database
	DropGroup	Removes an existing group from the database
	DropUser	Removes an existing user from the database
DtoLicenseMgr	GetProductInfo	Returns an XML formatted list of all Pervasive Software products found by License Manager

See the new methods in *Distributed Tuning Objects Guide*.

Product Authorization

Product authorization 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 authorization:

- Telephone Authorization
- Product Authorization for OEMs

Telephone Authorization

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

Telephone authorization 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 Authorization in *Pervasive PSQL User's Guide* for complete details.

Product Authorization for OEMs

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

- Product authorization information on the Pervasive Web site.
- 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. (Your Pervasive PSQL sales representative can provide more information about the Portal.) See also on the Portal:
  - *OEM Partner Handbook*, which has been extensively revised.
  - *Product Authorization for OEM Partners* white paper.
  - *Product Authorization Troubleshooting Guide for OEM Support Staff*.

Configuration Settings

Pervasive PSQL v11 includes changes to the following configuration settings:

- Communications Threads
- Listen IP Address

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*.

Listen IP Address

The Listen IP Address setting now accepts multiple IP addresses separated by a comma between each address. The string can be a combination of IPv4 and IPv6 addresses. Any of the IPv6 address formats supported can be used. See IPv6 Address Formats.

The Listen IP Address setting specifies the IP address or addresses the database engine listens on when TCP/IP Multihomed is Off. See also Listen IP Address and TCP/IP Multihomed in *Advanced Operations Guide*.

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 (32-bit Client DSNs now provide comparable functionality). See ODBC and Data Source Names (DSNs).
- DTI functions that manage 32-bit Engine DSNs. See DTI.

Pervasive Direct Access Components (PDAC)

The PDAC dynamic libraries for Delphi 2006 (and 2007 which is compatible with 2006) are still available in Pervasive PSQL v11 but will be removed from the product in the future.

Discontinued Features

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

- Support for Windows 2000
- Delphi and C++ Builder development environments version 6 and older. Pervasive PSQL v11 does not provide PDAC integration with development environments version 6 and older.
- The Pervasive PSQL ADO.NET Data Provider versions 2.1 and 3.0. The installation of the Pervasive PSQL v11 database engine automatically uninstalls either Provider if the version is detected.

Index