Pervasive PSQL v11

What's New in Pervasive PSQL

An Overview of New Features and Changed Behavior

PERVASIVE®

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

What's New in Pervasive PSQL
Candidate Release June 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 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 `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 Release Candidate includes the following new features and changes:

- Multi-core Support
- Relational Interface Support for 64-bit Architecture
- ODBC Interface Support for 64-bit Architecture
- IPv6 Support
- Product Activation
- 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 client/server 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 client/server 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 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.

But tuned in the right way, applications that would otherwise be slowed down can take advantage of multi-core processors and experience significant performance improvement. In many cases, swapping out portions of the application stack, such as the database, can address the issue 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. 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 recompile or rearchitect 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 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* 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 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. 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 has removed or diminished many bottlenecks in the database engine. 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.

Recall that the “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.

.

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 The relational support for 64-bit architecture is for Windows platforms only.

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.

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
Pervasive ODBC Engine Interface	Server 64-bit Server 32-bit Workgroup	• Creates 32-bit Engine DSNs • Connects to a local named database • Deprecated in Pervasive PSQL v11, as explained below
Pervasive ODBC Client Interface	Server 64-bit Server 32-bit Client 32-bit Workgroup	• Creates 32-bit Client DSNs • Connects to a local or remote named database or an Engine DSN • Interface GUI lists both named databases and Engine DSNs
Pervasive ODBC Interface	Server 64-bit Client 64-bit	• Creates 64-bit DSNs • Connects to a local or remote named database

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?	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. This conforms to industry norms.
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.
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. 64-bit Engine DSNs cannot be created with Pervasive PSQL.

Table 1-2 FAQs About ODBC and DSN Changes continued

Question	Answer
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?	No changes are required if the application uses DSN-less connections that connect using “Pervasive ODBC Client Interface.” If the application uses DSNs, you must create 64-bit Client DSNs that connect to a named database.
What about the DSNs for the Demodata sample database installed with the database engine?	If you install Pervasive PSQL Server 64-bit, both 32-bit and 64-bit Client DSNs are created for the sample database. 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.
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 Windows Registry and ODBC DSN Setup GUIs.
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?	No.
Are there any changes to existing utilities to support 64-bit ODBC and DSNs?	Yes. See Utilities Affected.

Table 1-2 FAQs About ODBC and DSN Changes continued

Question	Answer
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()

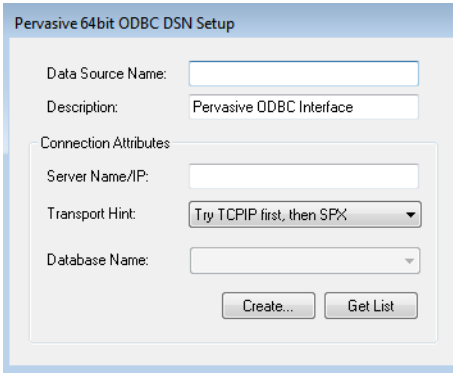
All of these functions operate only on the 32-bit registry. This applies even if a 32-bit database engine is installed on a 64-bit operating system. The 32-bit ODBC Administrator uses these DTI functions. Therefore, the list of existing Engine DSNs and newly created Engine DSNs are only for the 32-bit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

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

ODBC DSN Setup GUIs

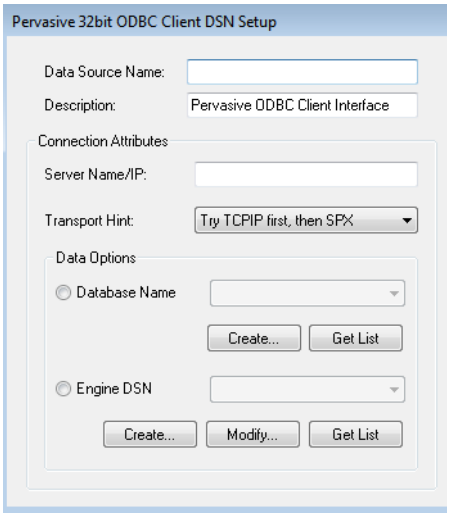
The following changes apply to setting up DSNs through ODBC Administrator.

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



- 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 1-1, Pervasive PSQL ODBC Drivers for Windows.



  - 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 lablel "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 lablel "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

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.

IPv6 Support

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

- Using Pervasive PSQL With IPv6
- Frequently Asked Questions
- IPv6 Aspects for Application Programmers

Using Pervasive PSQL With IPv6

Pervasive PSQL v11 supports IPv6 for the following access methods on Windows and Linux 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 can connect the following ways to a IPv6 host running Pervasive PSQL Server or Workgroup. These are the same connection methods as for IPv4.

Connection Method	Discussion
Name	The name of a server to which the client machine connects. Example: acctsvr1477
IP Address	The IP address of a server to which the client machine connects. See IPv6 Address Formats for the formats supported. Example: 1234:5678:90ab:cdef:1234:5678:90ab:cdef

Connection Method	Discussion
Uniform Resource Identifier (URI)	A URI is a string of characters used to identify a name or a resource. It may be a Uniform Resource Name (URN) or a Uniform Resource Locator (URL). Examples: btrv://User1@myhost/demodata?pwd=user1password btrv://myhost/demodata. See Restrictions. See also Database URIs in *Pervasive PSQL Programmer's Guide* for a complete discussion.
Universal Naming Convention (UNC) path	Pervasive PSQL correctly resolves UNC syntax regardless of the type of network operating system on the host server. The transactional interface expands mapped drives to a fully-qualified UNC file name (255 bytes limit). For instance, J:\Data\File.dat is converted to \\Servername\ShareName\Data\File.dat. Example: \\acctsvr01\acctspay\usinvoices\domestic.mkd See Restrictions. See also Universal Naming Convention (UNC) Path Formats in *Getting Started With Pervasive PSQL*.

Pervasive PSQL Licensing

Pervasive PSQL Server or Workgroup uses one user count for each unique incoming protocol from the same client computer. Multiple applications on a single client computer that use the same protocol are counted as one user, not separate users. For example, if two applications use TCP/IP and two other applications use SPX/IPX, two users are counted if all four applications run on the same machine.

If different address formats of the same protocol are used, only one user is counted. For example, if one application uses IPv4 and another uses IPv6, only one user is counted if both applications run on the same machine. IPv4 and IPv6 are just different address formats of TCP/IP.

IPv6 Address Formats

IPv6 addresses comprise 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 1-3 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. This format always begins with fe80 followed by 0:0:0:0:0:0:0. 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 has several 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.

Modifier	Explanation
/nn	Where nn represents an integer, used with a prefix to indicate how many bits are assigned to the prefix. For example, fe80::2b0/64 means that the first 64 bits define the prefix that starts with fe80 and ends with 2b0.
ipv6-literal.net	See IPv6-literal.net Names.

IPv6 and UNC Paths

UNC paths do not allow certain special characters, such as colons. Since 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 an IPv6 address with three changes:

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

Example:

Initial Address	fe80::713e:a426:d167:37ab%4
Modified Address	fe80--713e-a426-d167-37abs4.ipv6-literal.net

Ipv6-literal.net names are allowed in a URI or UNC used with Pervasive PSQL. The use of ipv6-literal.net names is also permitted in certain commands on Windows platforms, such as net use, which otherwise do not allow raw IPv6 addresses.

Bracketed IPv6 Addresses

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

Example:

Initial Address	fe80::713e:a426:d167:37ab%4
Modified Address	[fe80::713e:a426:d167:37ab%254]

Note that the ZoneID character “%” has been replaced with the escape characters “%25” for use with Pervasive PSQL. See Restrictions.

The use of square brackets is required for raw IPv6 addresses used in a URI or UNC with Pervasive PSQL. See Restrictions. Square brackets are also permitted in certain commands on Windows platforms, such as net use, which otherwise do not allow raw IPv6 addresses.

Utilities

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

Utility	See Also
Maintenance (and butil)	Manipulating Btrieve Data Files with Maintenance in *Advanced Operations Guide*
Monitor (and bmon)	Monitoring Database Resources in *Advanced Operations Guide*
Function Executor	Testing Btrieve Operations in *Advanced Operations Guide*
Rebuild (and rbldcli)	Converting Data Files in *Advanced Operations Guide*
bcfg	Configuration Reference in *Advanced Operations Guide*
dbmaint	Command Line Interface Utilities in *Pervasive PSQL User's Guide*
License Administrator (and clilcadm)	License Administration in *Pervasive PSQL User's 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.
Pervasive System Analyzer (PSA)	Pervasive System Analyzer (PSA) in *Pervasive PSQL User's Guide*

Restrictions

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

Table 1-4 IPv6 Restrictions With Pervasive PSQL

Restriction	Discussion
Square brackets are required for 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.[1] Examples: • btrv://czjones@[2010:b1::23]/demodata • btrv://abanderas@[2010:12:34:56:78:90:12:23]/demodata • \\[2010:12:34:56:78:90:12:23]\acctsvr1\Domestic\file.mkd Failure to bracket the IPv6 address results in status code 3014 or 3103, depending on the operation.
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 IPv5 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]
The Listen IP Address configuration setting can be set to only one address	The Listen IP Address does not accept a list of addresses. Only one address may be specified, which may be an IPv4 address format or one of the supported IPv6 address formats. See also Listen IP Address and TCP/IP Multihomed in *Advanced Operations Guide*.

Table 1-4 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.
[1] Pervasive PSQL does not officially support the use of a port number within a URI login. The database engine currently does not reject a URI with a port number, but be aware that such usage could change in a future release which would invalidate such URIs. If you do include a port number within a URI, only UNC-safe addresses are supported. The port number must be outside of the square brackets. For example, btrv://czjones@[2010:12:34:56:78:90:12:23]:1590/demodata. A port number is **not** supported for ipv6-literal.net addresses used in a URI.	

Frequently Asked Questions

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

Table 1-5 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)?	Not for Pervasive PSQL v11. Only the transactional and DTI access methods are supported.
Is IPv6 supported for the Macintosh OS X?	No.
Is IPv6 supported for Pervasive DataExchange, AuditMaster, and Backup Agent?	No.
Does Pervasive PSQL still support IPv6 if the operating system uses an IPv6 tunnel or router?	Yes.
How many user counts does Pervasive PSQL use per Client machine (or Terminal Server session) in a mixed network environment (an environment that includes both IPv4 and IPv6)?	See Pervasive PSQL Licensing.

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

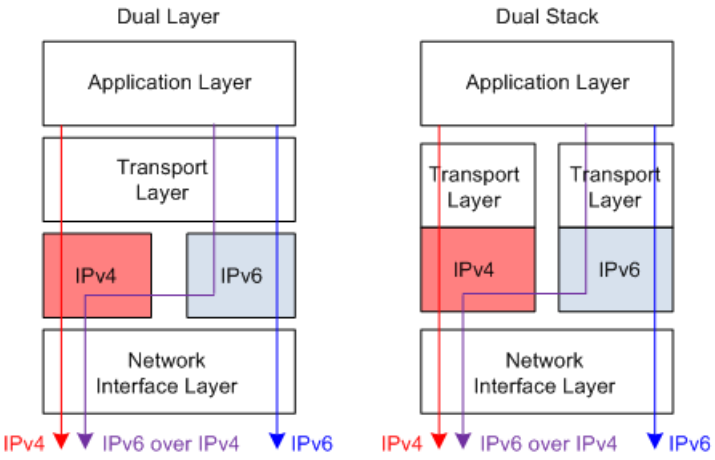
IPv6 is the next-generation Internet Protocol version designated as the successor to IPv4. IPv4 was the first implementation used in the Internet that is still in dominant use currently. Because of the age of IPv4, and the changing world environment of networking, IPv4 contains several limitations that 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 when IPv4 is phased out and IPv6 becomes its replacement, both protocols will be functional on certain operating system. 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

- 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

Dual Stack

- Available on Windows Server 2003, 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.
RedHat v4 and v5, and SuSE v10 and v11	Limited support for network GUI configuration, but the following command-line utilities are provided: ifconfig, ip, 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 filestypically 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 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 integer 10 is the ZoneID.

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

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*.

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.

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*.

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 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
- Delphi and C++ Builder development environments older than version 2006. Pervasive PSQL v11 does not provide PDAC integration with development environments older than version 2006.

Index