

# Pervasive<sup>®</sup> PSQL Xtreme I/O: Delivering Performance for 32-Bit Applications

---

A Pervasive Software White Paper

March 2008

## TABLE OF CONTENTS

<b>TECHNICAL SUMMARY</b> . . . . .	<b>3</b>
<b>XIO OVERVIEW</b> . . . . .	<b>3</b>
<b>CACHING</b> . . . . .	<b>4</b>
<b>INTELLIGENT COMPRESSION</b> . . . . .	<b>4</b>
<b>WRITING DATA</b> . . . . .	<b>5</b>
<b>SYMMETRIC MULTIPROCESSING (SMP)</b> . . . . .	<b>5</b>
<b>CUSTOMIZATION</b> . . . . .	<b>5</b>
<b>WHEN TO USE XIO</b> . . . . .	<b>5</b>
<b>HOW XIO INTEGRATES WITH PSQL v10</b> . . . . .	<b>6</b>
<b>XIO STATS</b> . . . . .	<b>8</b>
<b>XIO MANAGER</b> . . . . .	<b>8</b>
<b>CONCLUSION</b> . . . . .	<b>9</b>
CONTACT/TRADemark INFORMATION . . . . .	10

## TECHNICAL SUMMARY

Applications are often limited by their underlying hardware and operating systems. For example, disk-intensive applications such as databases may be limited by the speed of the underlying storage hardware (the RAM-DISK gap). Access to the storage subsystem is far slower than access to random access memory. In fact, we now describe storage access times in milliseconds while describing RAM access in nanoseconds.

A typical method of improving application performance is to add more memory to the system. This approach works but is limited by the operating system. For instance, Windows 2003 Server Standard Edition (x86) limits all applications to 2GB of addressable (virtual) memory, which incidentally includes the executable image or portions of it. Therefore, adding more memory will not necessarily translate into a direct boost to the application because the 2GB limit remains in force. Windows /3GB switch can overcome this limit and allows an application to use more memory. However, this approach is also limited in its benefit because it affects other parts of the operating system – creating less room for Windows and drivers.

A third approach to improving the performance is to install general purpose caching products. Again, such an approach is sub-optimal because caching products tend to cache everything; they are not design for a specific application and its data.

The last approach is to run the database application on a 64-bit version of Windows. Although this approach may enable the application to address more than 2GB of memory, the application is constrained by the performance and I/O of the operating system such as the paging of the application's data.

Another limitation is the available bandwidth between the CPU and RAM (the CPU-RAM gap) due to the amount of data transferred from main memory into the CPU's cache, consuming CPU-RAM bandwidth, and limiting system performance. Pervasive PSQL Xtreme I/O (XIO) seeks to eliminate the RAM-DISK and CPU-RAM) gaps by streamlining the I/O between the various hardware components.

## XIO OVERVIEW

Pervasive PSQL Xtreme I/O is a system accelerator exclusively dedicated to boosting the performance of Pervasive PSQL's database applications. It is implemented as a device driver for 32-bit (x86) versions of Microsoft Windows with a minimum of 2GB of RAM.

XIO uses several techniques to improve the flow of data between the server's storage, memory, and processor subsystems so they operate at maximum capacity. As an illustration, imagine that a Pervasive PSQL application is inserting a number of records into a table. This would result in random I/O requests to various regions of the database file that implement the table. XIO streamlines the requests to reduce the total number of I/O requests to storage, resulting in less load. Resulting in fewer loads on the storage controller and the PCI bus and by extension more bandwidth for other traffic.

Similarly, XIO improves the flow of data between main memory and the processor. Since data stored in memory is compressed, data is transferred by the processor across the Front Side Bus (FSB) and is uncompressed directly into the processor's L2 (or L3) cache. This is an effective use of the FSB's bandwidth. XIO with Pervasive PSQL allows only PSQL applications to benefit from those performance-enhancing techniques.

This white paper discusses the various techniques that XIO utilizes to increase the performance of Pervasive PSQL applications. For additional documentation on XIO, please see the [PSQL v10 Advanced Operations Guide](#).

## CACHING

Caching is a well-known approach to reducing the access time to data normally resident in storage such as a hard disk. Because cache is in DRAM, access to data is much faster. Given that XIO makes extensive use of caching and eliminates disk I/O for cached data, traffic on the PCI bus and to the storage controllers and hard drives is reduced. This reduction frees up resources to service other requests thereby reducing latency and increasing bandwidth.

Many caching subsystems are agnostic with respect to what is stored. In contrast, XIO stores only Pervasive PSQL database files. This puts all of XIO's cache at the service of the database engine. For example, an XIO cache of 1GB extends PSQL's cache by the same amount – with dedicated storage of Pervasive PSQL database files.

As mentioned earlier, 32-bit versions of Windows provide an application with only 2GB of addressable memory. Without XIO, the database engine's cache could never reach 2GB since the 2GB memory range also includes executable code segments. With XIO, the 2GB barrier is not an issue as the database engine cache extends with the XIO cache – up to the physical memory that Windows allows. For example, Windows 2000 Advanced Server (with /PAE) allows access up to 8GB of physical memory. With XIO, the database engine's cache is extended by whatever physical memory Windows 2000 makes available.

Using 64-bit addressing, XIO can access extended memory and can overcome the 4GB physical memory barrier on 32-bit systems. The net result is that 32-bit database applications, running on 32-bit operating systems, can now benefit from extended memory. The only constraint is that the operating system must make this memory available. (Some versions of Windows limit the amount of available memory to 4GB even if more is present. For more details on memory limits of various Windows releases, go to <http://msdn2.microsoft.com/en-gb/library/aa366778.aspx>).

Another aspect of the XIO cache is that it is dynamic. XIO monitors many aspects of system's performance (RAM and CPU usage, etc.), enabling XIO's cache to grow and shrink as it balances its need for RAM with rest of the system. This result is that an administrator is not required to choose a cache size. In addition, XIO is careful with its use of physical memory; XIO will not “keep” physical memory if it has no use for it. For instance, whenever a database application's data set can fit within the database engine's cache, XIO does not use any cache.

Lastly, the portion of the XIO that uses extended memory is fixed. In contrast to memory below the 4GB address boundary, XIO does not relinquish extended memory until it unloads. For example, consider a server with 8GB of physical memory and a version of Windows that supports 8GB of memory. With XIO installed, the database engine's cache extends to at least 4GB of RAM (from extended memory) or greater if the XIO detects a demand for its caching services.

## INTELLIGENT COMPRESSION

Intelligent compression plays a central role in XIO's cache. It extends the “reach” of the cache – its effectiveness vis-à-vis the data set it needs to cache. For example, with a 2-to-1 compression ratio XIO doubles the reach of the cache.

XIO contains a number of compression algorithms each of which is suited to a different type of data. Not only does compression extend the reach of the cache – it also reduces the burden on the FSB. For instance, with a 2-to-1 compression ratio, the cost of transferring a compressed buffer across the FSB is half the cost of transferring an uncompressed buffer. XIO has the ability to compress data at an 8-to-1 ratio.

Rather than applying one compressor for all the incoming data (i.e., one-size-fits-all), XIO applies a forward-looking algorithm on the data. This enables it to select the appropriate compressor – for the particular data - without trial-and-error.

## WRITING DATA

XIO also streamlines write operations to gain even more performance. It takes into account inefficiencies in current storage architectures and mitigates them via two techniques: write-aggregation and write ordering.

By collecting the results of many write requests and combining them into one larger request, XIO reduces the number of requests to storage, effectively reducing the load on the disk controller and the PCI bus. The result is improved bandwidth and decreased latency.

One of the biggest I/O performance limiters in current storage architectures is the movement of the hard drive's head as it moves to the correct location on the platter before it writes to the medium. The time it takes for the head to move from one location to another is called the seek time. The seek time directly affects the performance of a disk. One can imagine the head moving back and forth across the platter, and the time penalty associated with it, as unordered write operations go to the disk controller. XIO alleviates this problem by ordering the write requests to maximize the efficiency of the transfer.

It is important to stress that although XIO re-orders write requests for maximum throughput, it does not undermine the integrity of the data.

## SYMMETRIC MULTIPROCESSING (SMP)

A common problem in symmetric multiprocessing (SMP) architectures is CPU contention: the problem that occurs when multiple processors require access to the same block of physical memory. Although additional processors generally improve the performance of a server, the OS and applications need to be designed to take advantage of SMP architecture. Otherwise, the lack of synchronization among processors (for shared memory) reduces the benefit of this architecture. XIO is designed to take advantage of SMP platforms; XIO's internal structures can leverage the x86-processor architecture to gain even more throughput.

## CUSTOMIZATION

XIO can also be tailored for different application files. For example, if a database stores compressed images that do not yield a high compression ratio, it may be appropriate to disable XIO acceleration for the database file that holds the images. XIO includes an Exclusion Filter – a list of filenames and directories stored in a text file that are loaded by the database engine at startup time. An administrator can populate the list to instruct the engine to exclude specified names from XIO's acceleration, so those files will not be cached by XIO.

## WHEN TO USE XIO

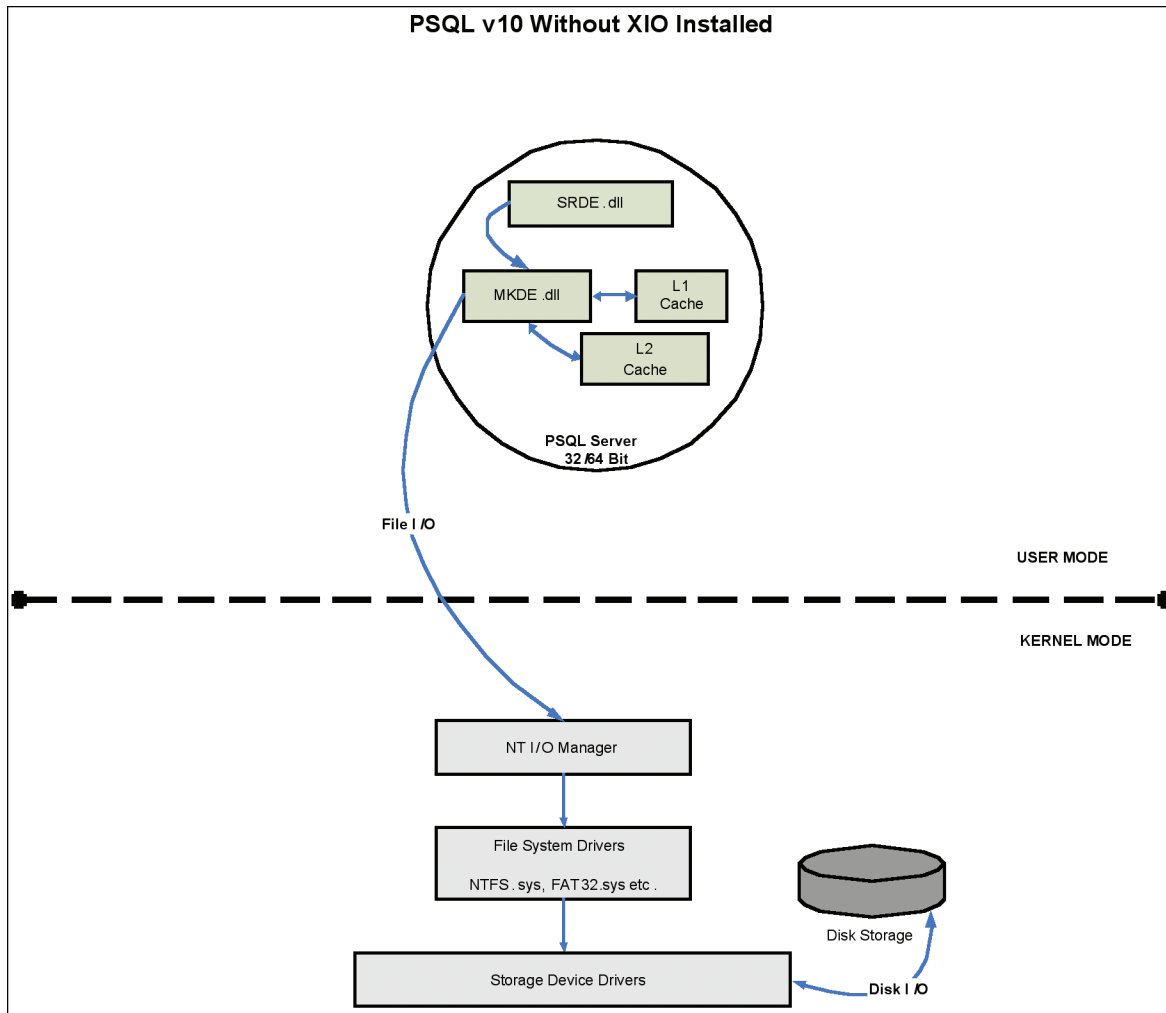
XIO is ideal for large databases that generate a lot of disk activity. For example when:

- The database size (data set) is larger than the combined size of PSQL's L1 cache and the Windows System Cache.
- The data request pattern is random (not sequential).
- The application is disk-I/O intensive; it generates a large number of disk requests.

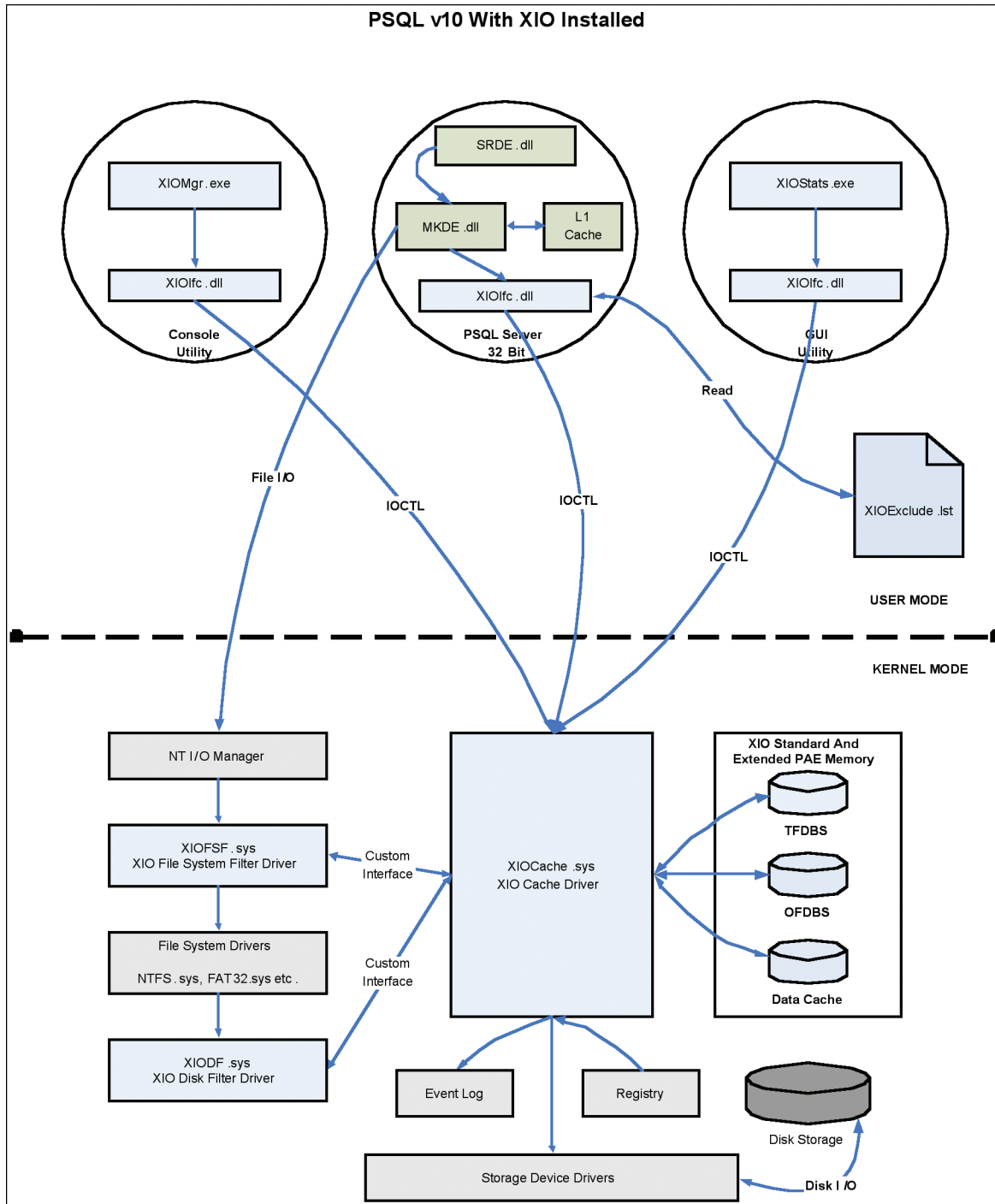
## How XIO INTEGRATES WITH PSQL v10

XIO installs with PSQL at the user and kernel mode OS layers. At the user mode level, two utilities (xiomgr.exe, xiostats.exe) and an interface module (xioifc.dll) provide a common view and control interface into XIO. The two diagrams below show PSQL without and with XIO installed.

Without XIO installed, PSQL uses an L1 and L2 cache. The L2 cache uses compression to extend the amount of data stored but the Windows 2GB limit is still present. In a standard 32-bit install of PSQL, the L1 cache will consume 20% of available memory and the L2 cache can consume up to 60% of available memory.

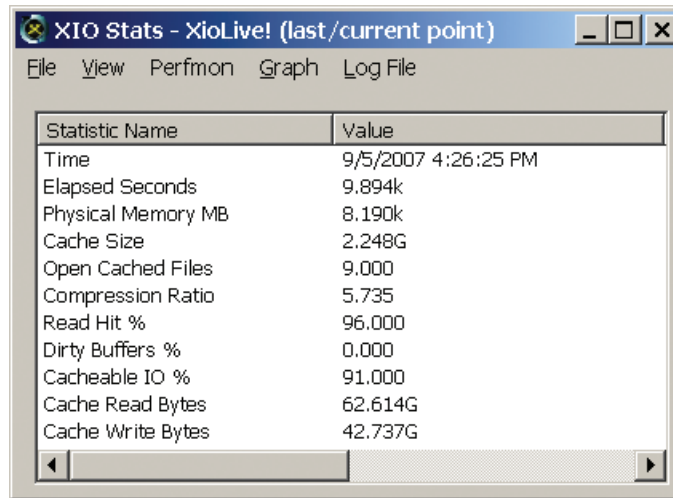


With XIO installed, the standard PSQL L2 cache is “turned off” to allow XIO to assume the role of PSQL L2 cache and provide more memory and faster compression. Notice the linkages via the interface module at user mode level and the three additional XIO drivers (XIO File System Filter, XIO Disk Filter, and XIO Cache) at the kernel mode level. The OS registry setting “large system cache” gets turned off to give XIO even more system memory dedicated exclusively to PSQL database files.



## XIO STATS

XIO Stats is a GUI application that displays real-time statistical information about the XIO cache parameters and key performance values. The utility provides basic statistical information, is for advanced users, and requires an understanding of the various statistical elements and values. Information appears as a matrix in the main window of the utility:

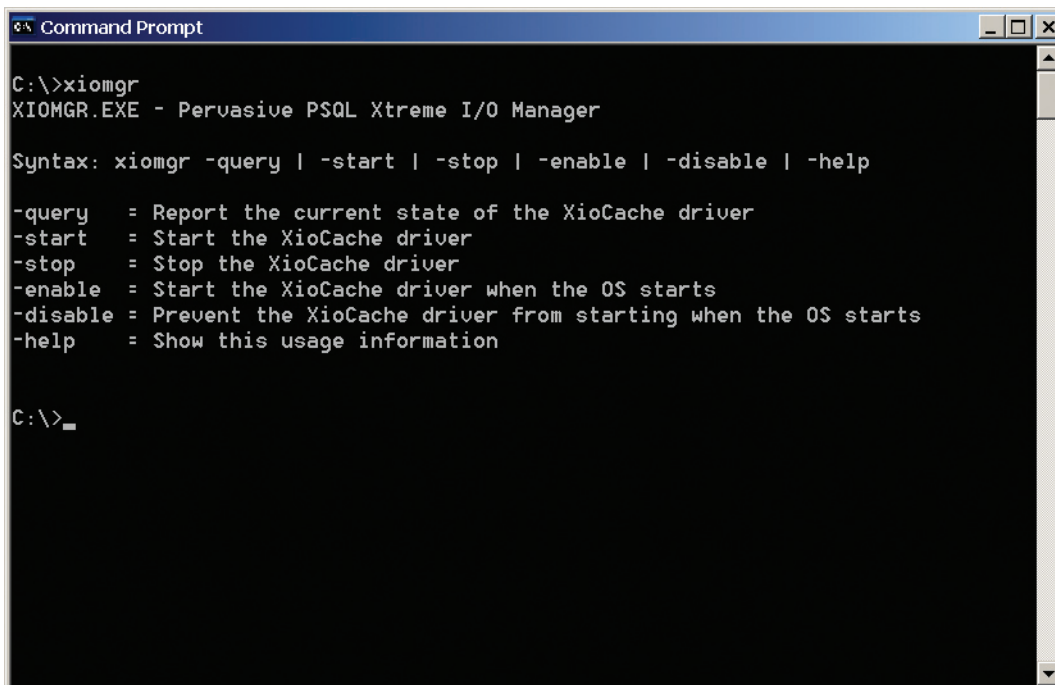


The screenshot shows a window titled "XIO Stats - XioLive! (last/current point)". The window has a menu bar with "File", "View", "Perfmon", "Graph", and "Log File". The main content is a table with two columns: "Statistic Name" and "Value".

Statistic Name	Value
Time	9/5/2007 4:26:25 PM
Elapsed Seconds	9.894k
Physical Memory MB	8.190k
Cache Size	2.248G
Open Cached Files	9.000
Compression Ratio	5.735
Read Hit %	96.000
Dirty Buffers %	0.000
Cacheable IO %	91.000
Cache Read Bytes	62.614G
Cache Write Bytes	42.737G

## XIO MANAGER

XIO Manager is a console application (xiomgr.exe) that controls the start mode and status of the XIO driver. This allows the driver state to be changed for maintenance or troubleshooting. XIO Manager supports the following command switches:



```
C:\>xiomgr
XIOMGR.EXE - Pervasive PSQL Xtreme I/O Manager

Syntax: xiomgr -query | -start | -stop | -enable | -disable | -help

-query   = Report the current state of the XioCache driver
-start   = Start the XioCache driver
-stop    = Stop the XioCache driver
-enable  = Start the XioCache driver when the OS starts
-disable = Prevent the XioCache driver from starting when the OS starts
-help    = Show this usage information

C:\>_
```

## CONCLUSION

XIO is exclusively dedicated to boosting the performance of Pervasive PSQL's database 32-bit based applications. It is implemented as a device driver for 32-bit (x86) versions of Microsoft Windows operating systems with a minimum of 2GB of RAM.

This sophisticated I/O accelerator extends memory and increases the performance capabilities of the Pervasive PSQL Summit v10™ database engine. It uses a variety of techniques and algorithms to achieve performance gains for large database applications that generate random database requests and can benefit from having the data loaded in physical memory for faster access.

Learn more about the [Pervasive PSQL](#) database and Pervasive's Security Solutions: [AuditMaster](#), [Backup Agent](#), and [DataExchange](#).

## **Contact Information**

Pervasive Software Inc.  
12365 Riata Trace Parkway, Building II  
Austin, Texas 78727

### United States

800.287.4383  
512.231.6000  
Fax: 512.231.6010  
info@pervasive.com

### EMEA

+800.1212.3434  
cic@pervasive.com

<http://www.pervasive.com>

---

© 2008 Pervasive Software Inc. All rights reserved. All Pervasive brand and product names are trademarks or registered trademarks of Pervasive Software Inc. in the United States and other countries. All other marks are the property of their respective owners.