Pervasive.SQL 2000 Backward Compatibility

A White Paper From

GOLDSTAR SOFTWARE
www.GoldstarSoftware.com

For more information, see our web site at
http://www.goldstarsoftware.com

Pervasive.SQL 2000 Backward Compatibility

Last Updated: 09/18/2000

Pervasive Software makes the claim that Pervasive.SQL 2000 is backward compatible for all older applications, especially those running Btrieve 6.15. This paper examines this claim in more detail, showing the highlights (and the lowlights) of the compatibility issues. The information provided herein is the result of our own experiences. ***Since every application and network is different, we urge proper testing of your application in your environment!***

Types of Compatibility

There are several different ways in which we can examine the level of backward compatibility. Some of these obviously overlap, while others are categories onto themselves. We'll examine the most common Btrieve Applications, as well as the less common Scalable SQL Applications and ODBC Applications. Finally, we look at two options, upgrading just the server and upgrading just the clients. Both of these are key to Migrating to Pervasive.SQL 2000. However, since these sections build on previous material, it is important to understand the other sections as well.

The simplest question, and the reason you are probably here reading this, is "Will my application run properly on Pervasive.SQL 2000?". Unfortunately, the answer to this one is not so simple. It is always best to check with the application developer directly for their current level of support. This is especially important for patch releases. Outside of this information, there are quite a few types of applications to consider and issues raised during the process, so we'll try to break it down into components as best as we can.

Btrieve Applications

Due to the intentional backward compatibility which Pervasive strives to maintain, one can say that, in general, *any properly-written Btrieve application should be fully compatible with the Pervasive.SQL 2000 database engine*.

Now, let's dissect that statement into the good and bad points.

1. "properly-written": A Btrieve application developer has a few ground rules to follow when using the Btrieve API, and these rules have been around forever, which is what guarantees the backward compatibility for their applications. If any of the rules have been broken, the application may be restricted to ONLY running on the older software and will preclude any upgrades.
    a. One such rule is the proper termination of filenames. The spec indicates that all filenames (on File Open calls) must be terminated with a space or a NULL byte. Some older applications, especially Btrieve 5.x for DOS applications, took advantage of the fact that Btrieve ignored all characters after the 3-character filename extension. These apps have been having problems on Btrieve 6.15 and 7.x for years, but do still exist. Consequently, storing Btrieve files in directories with embedded spaces in them is bad. Caveat: A recent change to the 7.x Btrieve API allows the use of filenames with embedded spaces. However, this extension is enabled on a per-computer basis, not on a per-application basis. As such, developers who change the requesters to require a null terminator for their app can often break other applications (those that use a space terminator) running in the end-user environment.
    b. Another rule is the definition of unused parameters for a function call. The Btrieve API has always indicated that any unused parameter for a function should be initialized with a zero value or a pointer to a null byte. This allows the API to grow while still maintaining backward compatibility. Some developers take shortcuts in this process, which then break the applications when the API changes. An example of this is the STAT call. Recently, Pervasive added a special internal version of the STAT call which used a key number of -2. Some applications which did not initialize the key number for a STAT call would actually end of with an incorrect data buffer (and thus incorrect results) by not setting this value to 0 before the call.
    c. Some developers tried to work around old bugs or by exploiting undocumented features or undefined functions. By doing this, if the bugs are fixed in future revisions, or if the functionality of the "feature" is eventually changed or defined differently, the applications break.

2. "Btrieve application": A Btrieve application is on which exclusively accesses the database files through the Btrieve API, and ONLY through the Btrieve API. There are some applications which violate this rule quite completely.
    a. ODBC applications: Any access via ODBC is suspect for several reasons. Primarily, the entire ODBC access mechanism has been rewritten from the ground up, and there is no compatibility provided here. See the ODBC section below for better details.
    b. NetWare SQL or Scalable SQL applications: Applications built on the XQLM or XQLP interfaces, such as Xtrieve, may or may not work correctly. Ultimately, the XQL calls are translated into Btrieve calls, but the older XQL drivers may not have followed correct rules in all cases, or may not even utilize documented functions in all cases. Since the Scalable SQL programming API's have been completely removed from Pervasive.SQL 2000, any applications using it may be seriously restricted from upgrading.
    c. OS-Level Data File Access: Some applications access Btrieve data files at the operating system level. This is much more subtle, but just as much of a danger to be avoided. One example is PeachTree, which tries to open the file at the OS-level to see if anyone else has it locked. (This is a very bad application design concept, and should be avoided by developers at all costs.) Another more common example is the application vendor who has been kind and included a database rebuild and/or recovery facility built into the application. With Btrieve 6.x, the logical database file is stored in a single physical file, so OS-level was OK. However, the 7.x file format can split a large logical file into as many as 32 pieces. Applications which do not expect this may actually corrupt the database beyond repair during the rebuild process. This can often be avoided by leaving the files in 6.x format, but then you are still restricted to 4GB file sizes.
3. "should be": There are always exceptions, new problems created by bug fixes and feature updates, and such. A prime example is the first release of SP2 which broke GetNextExtended operations which affected several major applications. It is ALWAYS a good idea to bring the system up on a test server and have a few users bang against it for a few days prior to a major roll out to a production environment.

With that out of the way, let’s look at some specifics on the platforms for DOS and Windows-based applications.

DOS-Based applications
DOS-based applications are VERY simple, in that they use a simple DOS engine (BTRIEVE) or DOS requester (usually BREQUEST). If you are using either of these programs and ONLY change the server engine to Pervasive.SQL 2000, you should experience no problems. However, it is important to upgrade to at least BREQUEST v7.00.201 to get some bug fixes which appear on Windows NT systems. There is also a newer version to be released sometime soon (7.00.400?) which will provide some performance enhancements when logged into NetWare 5.x servers with TCP/IP connections.

The complications come in when you install the Pervasive.SQL 2000 components on a workstation. The installation installs a series of components which comprise the DOS-box requester, also known collectively as BTRBOX. When the BTRBOX95.VXD file is installed to a workstation and loaded, it intercepts all of the older DOS TSR interrupts (0x7B) for itself, and tries to pass them off to the BTRBOX95.EXE application. If this is NOT running, a status 20 will result. Note that this occurs *even if BREQUEST is loaded*! You have two options to work around this issue.

- Configure the Win32 requesters correctly to provide access through the new components. This will provide full TCP/IP capabilities, and better performance in some cases. This DOES work to older 6.15 engines, and is the ONLY way to access Pervasive.SQL 2000 **local** engines from a DOS application. This is the preferred mechanism, but may not be desirable due to bugs, incompatibilities, or performance issues -- you will need to test your own applications. Why are there problems? In the old DOS world, the app had full control over the computer, and passing a bad buffer wasn't necessarily a bad thing. However, in the Windows world, this can cause GPF's, Access Violations, and worse. As a result, the Windows-based code must adhere to the more rigid requirements of app development -- clean code. Another problem in versions prior to SP2 was that the BTRBOX95.EXE application had to be manually started by the user. It then sits on the task bar and can be erroneously shut down by the user if not properly trained. With the release of Service Pack 2a, this icon now sits in the system tray and is far less likely to be terminated by the user incorrectly.

- Remove the BTRBOX requesters and continue to use BREQUEST or BTRIEVE. Check the Control Panel's Add/Remove Programs applet and remove the DOSBOX support. It is also possible to temporarily remove it as a test -- just rename BTRBOX95.VXD and reboot the workstation. It will report an error about a missing VXD, but this is OK for a temporary test. It can be completely removed either via the Control Panel or by manually editing the registry.

Windows-Based Applications
Most Windows-based applications, either 16-bit or 32-bit, will run without problems on Pervasive.SQL 2000. They can continue to use their existing DLL requesters, or they can use the newer Pervasive.SQL 2000 requesters with no major issues. Some versions of newer requesters may be more buggy, but they typically work better and faster than their older counterparts. Typical problems with the Win32 requesters from Btrieve 6.15 include random or recurring Status 94 problems -- these all but go away in the new versions and are recommended for anyone running in an NDS environment.

Scalable SQL Applications

Scalable SQL, and the NetWare SQL product before it, are the older SQL engines in the product line. There is a high level of backward compatibility from NetWare SQL 2.11, NetWare SQL 3.00 and Scalable SQL 3.01. (The Btrieve 6.15 ODBC drivers are really a Scalable SQL application.) Then, in Pervasive.SQL 7, the SSQL engine was rewritten as version 4, which introduced some sweeping changes. The most critical changes are:

- Better ANSI Compatibility
- Full Support for Btrieve 7.x
- A Cost-Based Optimizer (instead of the older Syntax-based)

These changes, especially those to the parser, made some older DDF's unusable, since they contained reserved words (which were now prohibited), invalid characters (such as %, $, #, etc.) in field and table names and other problems (first character is a digit). As such, the compatibility of the DDF's themselves went down by quite a bit.

However, Pervasive did try to retain the backward compatibility at the application programming interface (API) level. In fact, the XQLM interface (which used SQL-like commands) was mostly untouched and still worked well. However, the XQLP (primitive-level commands) interface were deprecated and scheduled for removal. This did cause some problems with XQLP applications.

So, what does all this old stuff have to do with Pervasive.SQL 2000? The older SSQL engine was known to be very slow and it had a lot of "garbage" -- with patches, workaround, etc. built into it. The general architecture of the engine was considered mostly unstable, and a newer, faster engine was needed to provide better ODBC support. So, they eliminated all traces of the older Scalable SQL technology from the engine and introduced a whole new SQL engine, called the SRDE. Unfortunately, this new engine had NO available API (except ODBC), so **ANY program written to Scalable SQL will not work on Pervasive.SQL 2000**! As such, there is NO backward compatibility for these apps.

ODBC Applications

ODBC drivers have been available for Btrieve for a while now. Originally, there were drivers for MS Access, which were very poor. The 1.0 drivers from Pervasive were not much better. The last official release of the ODBC drivers for Btrieve 6.15 is version 2.04. As mentioned above, these drivers actually call the Scalable SQL engine -- either on the server or on the workstation. If no server engine exists, the workstation engine is used, which then makes Btrieve calls to the database engine. In other words, the applications makes ODBC calls on the workstation which makes SSQL calls on the workstation which makes Btrieve calls to the server.

When Pervasive.SQL 7 was released, it shipped with a Scalable SQL 4 engine in the box which ran on the server, providing better performance for SSQL and ODBC users. The newer versions of the 7.x drivers are actually v2.54, and will ONLY work with a server-based Scalable SQL 4 engine. (Actually, the original PSQL7 ODBC drivers through 2.52 did still ship with the workstation SSQL 3.01 engine as part of it, but it was removed with the release of the SP3 client.) In other words, the applications makes ODBC calls on the workstation which makes SSQL calls to the server which makes Btrieve calls on the server.

The newest ODBC drivers, shipping with Pervasive.SQL 2000, embody a whole new philosophy. Instead of turning the ODBC request into a Scalable SQL request, the ODBC request is sent to the server for processing. In other words, the applications makes ODBC calls to the server which makes Btrieve calls on the server. This results in much better performance and should result in lower network traffic as well.

This architecture and history is important because it lets you know what will work. Let's look at a few examples:

- If you install the newest ODBC drivers to a workstation (which install as part of the client installation), you will ONLY be able to access Pervasive.SQL 2000 servers via ODBC from that workstation. Any Btrieve 6.15 or Pervasive.SQL 7 servers will be inaccessible using the newer drivers.
- If you have an older v2.54 ODBC driver from Pervasive.SQL 7, it will be able to access Pervasive.SQL 7 servers only. Since the Pervasive.SQL 2000 server install does not provide the Scalable SQL for the server, and since the v2.54 ODBC client does not provide the SSQL engine for the workstation, you might be stuck. (There IS an upgrade path, see below.)
- If you have an older driver than the above, you will be able to access both Btrieve 6.15 servers (with the workstation SSQL engine) and Pervasive.SQL 7 (with the server SSQL engine). In addition, you will also be able to access Pervasive.SQL 2000 engines! Why? Because the workstation SSQL engine makes simple Btrieve calls at the lower level, and appears as a simple Btrieve application to Pervasive.SQL 2000! You won't get any of the performance gains, but you'll have some backward compatibility for a while, at least.

It IS possible to have both the 6.15 and PSQL2000 ODBC drivers installed on the same machine at the same time. In fact, this does seem to work well from most workstations. Whenever possible, however, try to have the 6.15 ODBC drivers use a 6.15 WBTRV32.DLL, as this will avoid some problems seen in some applications, most notably Crystal Reports.

Upgrading Pervasive.SQL 7 to Pervasive.SQL 2000 to Maintain Scalable SQL Access

One rarely-seen configuration is having the Scalable SQL 4 engine (from Pervasive.SQL 7) and the new SRDE (from Pervasive.SQL 2000) running at the same time. The ONLY way to get this to work is to install (and completely patch) the Pervasive.SQL 7 engine first, then install & patch the Pervasive.SQL 2000 engine. When this occurs, you will be able to access both the older and newer relational engines at the same time. However, there are a few notes about this:

- Once you install Pervasive.SQL 2000, you can never install any newer patches for Pervasive.SQL 7. This goes for both the workstation and the server.
- With two SQL engines, additional resources will be needed on the server.
- Although it should work just fine, this is not an officially tested scenario, and some problems may arise which cannot be fixed.

Planning a Migration

A migration from an older engine to Pervasive.SQL 2000 is easy, but consideration should be made to the three mechanisms of completing this task:

Upgrading the Server Engine First

This is typically an easy one. Most Btrieve-based applications will have no problems with this, as the older requesters should still work just fine with the newer engine. This is the easiest way to implement the change, since it guarantees that you'll have a working server engine before touching a single workstation. If problems arise in the server engine, it can easily be removed and backed out, given a proper backup. Finally, since the Pervasive.SQL 2000 server installation also installs the Client Installation files, and since these will likely need to be patched prior to installation anyway, it is much easier to migrate by doing the server first.

Upgrading the Client Requesters First
However, Pervasive officially recommends doing the client first, since newer client requesters are tested with older server engines, but not vice versa. However, as the ODBC section above indicates, problems with the ODBC side can arise if you are not careful. The only other problem is trying to obtain the latest patched client installation, so that you don't have to install the older client & then apply a patch right away.

Upgrading All Components at the Same Time
Can you do everything at once? Certainly! This may even be required for some combinations of applications and/or workstation components, and is by far the fastest way to get a system upgraded. However, doing a wholesale in-place upgrade is inherently dangerous since it is very difficult to roll back to the older software should something be found which is completely broken. I always avoid this option unless I've fully tested the application in a like environment first.

So, which do you do? You must decide for yourself based on the environment, time allotted, and risk factors. Personally, I usually do the server installation first, since it results in a quicker overall installation and it parallels a new installation more closely. A word of advice -- never proceed without a backup and a back-out plan!