Btrieve Data File Maintenance

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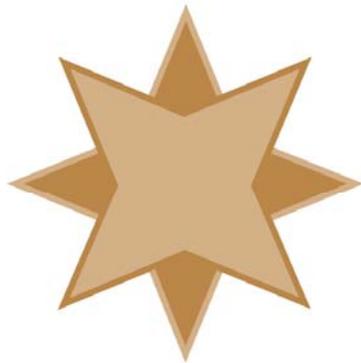
For more information, see our web site at
http://www.goldstarsoftware.com

Btrieve Data File Maintenance
Last Updated 04/07/2010

Btrieve data files do not require periodic maintenance, unlike some of the more administration-intensive database products out there. In fact, the majority of Btrieve files sit happily for their entire lives without ever having to be touched outside of the program that creates and accesses them. As a result, using BUTIL or some of the other utilities is relegated to the programmers in the world, since these are the people who can benefit most by their use.

In fact, there are only two cases where file maintenance for a normally-running Btrieve database may be required.

First, if the developer makes a change to the application that requires changing the size of the record in a file, a manual BUTIL rebuild process may be utilized. Oftentimes, the developer will create a version upgrade utility, or some other process to perform this function automatically. If this is done from the workstation, however, a long time may be needed to convert large files. It may be desirable to manually perform the process from the server instead.

Second, if a database is very large and the application purges (deletes) a large number of records, you may wish to give some of the newly freed up storage space back to the operating system. This is important because Btrieve does not release empty data pages back to the operating system -- it just marks them as free pages for future use by the engine. As a result, if you take a 1GB file and delete all of its records, you will still have a 1GB file when you are done (albeit one with a lot of free space). In this case, the rebuild process can be used to trim that file down to a few KB. If your file is expected to grow again in to the original size in the near future, however, rebuilding the file will make no sense, since the engine will just have to reallocate the disk space again.

Beyond these two reasons, the most important other use of BUTIL is to recover information from a damaged file. When problems occur, such as workstation crashes (using the workstation engine), file server crashes, network failures, power outages, bad software, etc., you may need to manually recover one or more database files, and this utility is your first key to doing so.

Common Causes of Data Corruption
Data corruption can occur. In some environments, the dreaded "Btrieve Status 2" error message occurs once or twice a year, in some it will never occur. Still other environments seem to be plagued by data corruption to the tune of once a week or more. What can be done?

The best way to avoid data corruption is to eliminate its causes. (Sounds simple, huh?) Here are some common causes of corruption that you can avoid:

1. **Pervasive Engine**: Make sure that you are running the latest, patched version for your environment. If you are running the client (i.e. workstation) engine in a multi-user configuration, strongly consider using the server engine instead. It may cost more and be slightly harder to set up, but it will pay for itself in increased performance and improved stability.

2. **Data File Format**: If you are running Btrieve 6.x or higher, then your files should be in the 6.x format (unless you must be able to access the data from a 5.x engine). Since the 6.x format has better error recovery, leaving them in 5.x format will also hurt performance and stability. If you are running Pervasive.SQL V8 or higher, you will have another performance gain on database writes by upgrading the files to the 8.x format or higher.

3. **Network Layer**: Make sure that your physical network is up to par, and that you are using all of the latest client software and LAN drivers.

4. **Network OS**: Make sure that your network operating system is fully patched with all recommended patches from the vendor.

5. **Hardware**: A failing NIC, memory, hard drive, and controller can also cause corruption, especially if the network OS doesn't detect the failure. These can cause bad information to be written to the drive, corrupting the file for you. Often, again, bad drivers can be the culprit. Stay on top of your vendors for all the latest stuff.

6. **Power**: Bad electrical power is really the most common problem, and it corrupts more environments than you know. How many people have a small heater or warmer for their coffee cup plugged into the same outlet as their computer? Laser printers on the same circuit can also cause problems, and any electrical problems such as floating grounds can downright destroy entire racks of equipment. If you constantly see problems, put EVERYTHING on its own UPS – the server(s), all network hubs, and each workstation.

Data Corruption – Real or Phantom? Only the Shadow Knows…
The first thing to realize is that NOT all status 2 reports are really corrupted files, and that each environment has a potential for reporting a status 2 on data which is still intact and fully readable. When this happens, save yourself some time and eliminate phantom corruption before starting a recovery process.

NetWare Phantom corruption
In the NetWare 3.x world, phantom corruption is caused most often by bad networking drivers and bad OS patches being applied. Make sure that only the necessary patches are installed, and that anything which is not recommended by Novell is NOT running.

Unfortunately, this phantom corruption can quickly lead to real corruption, so vigilance is the only defense.

Changes to the disk system in NetWare 4.x (and above, through at least 6.5) can cause problems. One particular phantom corruption problem affects NetWare 4.x and 5.x servers, called the TURBO FAT CACHE bug. The Turbo FAT is a cached file allocation table built by NetWare when the physical FAT exceeds 64 entries. (The actual size of the files affected by this issue is determined by block size and other factors.) The Turbo FAT is used by the OS to quickly locate any given portion of a file without having to read the entire FAT chain, and it is cached in memory between file accesses.

What really happens? Because Btrieve for NetWare uses Direct File Access (DFA), it goes direct to the disk for much of its disk I/O. Over time, this circumventing of the OS can cause the Turbo FAT in memory to become invalid, pointing to the wrong data blocks on disk. Then, when the system is reading data, it detects the corruption and reports the Status 2, since it could not read the information it wanted to read.

To properly detect this phantom corruption, use the DOS COPY function to copy the file from the NetWare volume to another volume, server, or workstation hard drive. This will read the file from the current FAT entries and properly copy it. Then, copy the file back to the original location and retry the operation. If the problem goes away, then you have the phantom corruption. Essentially, you must clear the Turbo FAT cache to clear the corruption, so dismounting and mounting the volume, downing and restarting the server, or any other flushing of cache will work as well

Important: DO NOT USE NCOPY! Many of you recognize that NCOPY is much faster when you are copying files within a server. However, NCOPY will access the Turbo FAT when reading the data. The Turbo FAT points to the wrong blocks – this can make your copied file a true corrupted file!

How do we prevent it? Luckily, Novell is aware of this problem, which is actually a problem associated with DFA that will affect other DFA apps (like Oracle) as well. They have a patch available, called TURBODIS.NLM, which can be loaded to completely disable the Turbo FAT Cache. The actual performance penalty on the server is negligible. Download the file TURBOD2.EXE from Novell's website and apply this to the server following the directions in the README file.

Windows Server Phantom Corruption
With the Windows Server engine, the entire caching algorithm is different, so we don't need to worry about the Turbo FAT problem. Unfortunately, there are other problems we get to worry about!

The Btrieve engine for Windows has an option to use the System Cache, which is the default. This means that the information being read into the Btrieve cache is first cached by the Windows NT OS, and then cached again by Btrieve. At first, this seems inefficient, but the engine was written to take advantage of this fact.

The problem comes up if you inadvertently turn this option off, or if you need to disable it for some other reasons. (In particular, the 6.15.445 engine for Windows NT has some known problems with files over 2GB that requires that this be turned off to avoid corruption.) In this case, the problem is actually that the OS may be modifying the files without informing the Btrieve engine.

What really happens? Btrieve for Windows uses a *persistent* cache. This means that file data that is in the cache at a given time will still remain in the cache, even if the file is closed and then reopened. This provides a performance enhancement for files that are opened and closed repeatedly by an application, but it also brings up an interesting dilemma – what happens if a file is completely closed, and then replaced with a different version of the same file?

To see this problem, the engine must be configured to NOT release resources, or the time interval in question must be less than the specified release time (30 seconds by default). This ensures that the file information is still cached and in memory while the file is closed. When the file is replaced with a different copy and an application opens and reads the file, the cached FCR pages will be used to start the read operation, along with (possibly) cached index pages. Unfortunately, if the new data file is sufficiently different, any page reads which need to access the physical file will get confused, since the expected pages may not be available, and the engine reports a Status 2. Shutting down the engine and restarting it clears the cache, and the file can be accessed with no problems.

How do we prevent it? Unfortunately, there's no easy patch to fix this. Only diligence on the part of the system administrator or users can prevent it. In essence, if ANY Btrieve file is being accessed by an application and must be replaced by OS-level file calls, the engine should be shut down before allowing users to access the engine again. This can be an expensive proposition, especially in a 24x7 environment, but is necessary.

Other OS Errors Masquerading as Corruption
In some cases, other, OS-related issues arise which cause the database engine to get confused. For example, a server with an iSCSI connection that loses its primary network link suddenly has no access to the disks, and it reports an I/O error as a result. When you have corruption, you should FIRST review the server's PVSW.LOG file and look for any "System Error" messages in the log. The THIRD number in the error message if the OS error, and if it is non-zero, then you should check with the OS vendor for information on this error code.

Real Corruption
Well, if you've checked the above, then you're probably one of the lucky folks who has real corruption. In this case, part of the file has been damaged in such a way that you cannot retrieve the information properly, and you have to rebuild it.

KeyCheck: Confirming Corruption

The KeyCheck utility from Goldstar Software (www.goldstarsoftware.com/tools.asp) can provide a simple way to check your database files for corruption via a free DOS version, or via a free 30-day trial on the Win32 version. KeyCheck reads every record of a given database file via every index, and tries to find any problems when doing these reads. A simple command would look like this:

```
C:\Data>keycheck MYFILE.BTR /s
KeyCheck Version 3.21: 03/31 (C)2009 Goldstar Software Inc.
Registered to Goldstar Software Inc. (Site License GS)
=============================================================
KeyCheck Report for MYFILE.BTR...
=============================================================
Press <Esc> to Abort...
Indicated Record Count = 317295
Processing Key # -1...File OK After 317295 Records.
Processing Key #  0...File OK After 317295 Records.
Processing Key #  1...File OK After 317295 Records.
Processing Key #  2...File OK After 317295 Records.

Scanning completed.
Maximum Status: 0: File OK
```

Registered users of KeyCheck can also use the Quick-Read and the Multi-threaded options to vastly improve the performance of the checking process.

BUTIL: Man or Mouse?

The BUTIL program is one of those small, all-purpose utilities that every Pervasive database user should know how to use. You can use it to recover data from corrupted files, or just to rebuild a file to recover disk space.

You can liken BUTIL to a spare tire – a lot of people have it, but few know how to actually use it **quickly** when they need to. Often, in the race to get a crashed system up and running again, it will be those people most familiar with this tool who will be back on the road in a relatively short time. The others are doomed to fumble around in the darkness for a while.

Common BUTIL Functions

The BUTIL program has several common functions that you'll use as you support Btrieve applications. There are also additional functions that are used mainly by developers, which we won't go into here.

You can get information about the options and the command syntax by typing BUTIL by itself (LOAD BUTIL for NetWare). If you need the syntax for a single command only, you can enter "BUTIL –CLONE", which will give you the proper syntax for just the CLONE option.

The most commonly used functions in BUTIL are:

STAT

The STAT option is used to retrieve information from the header (File Control Record) of a Btrieve file. The information returned back from a STAT report is essential to understanding and knowing how much data was lost after corruption occurs. This data includes:
- File Info, such as Page Size, File Flags, Number of Records, etc.
- Index Info, such as number of keys, data type, size, and number of unique records.
- Much more...

The most important statistic to know is the number of records in the file, so that you'll know if your recovery process was successful or not.

RECOVER/SAVE

The BUTIL –RECOVER process and the BUTIL –SAVE process are very similar.

When performing a SAVE, the file information is read based on the specified key (Key 0 is the default if none is specified) and stored into an "unformatted" ASCII file, explained below. The common format for this command is:

> **BUTIL –SAVE <BtrieveFile> <UnformattedFile>**

A RECOVER on a file is very valuable in cases where one or more index records or pages have become damaged, and the SAVE operation fails. Instead of reading the file based on the index information, a RECOVER reads through the file in physical record order, from start to finish. If the corruption is exclusively in index pages, this process can often recover all of the records from a damaged file, quickly and easily.

> **BUTIL –RECOVER <BtrieveFile> <UnformattedFile>**

You may also wish to use the /I and /J options, which cause BUTIL to start at the beginning and work until it hits an error, then it jumps to the end and works backwards. Unfortunately, if multiple blocks in the middle of the file are completely obliterated, even a RECOVER may fail, necessitating the use of more powerful tools, which will be discussed below.

What is an "Unformatted" file?

Perhaps "Unformatted" is not the correct name – since it does indeed have a format. This could also be considered to be a "Raw" format. An unformatted file is a binary file with contains records from a Btrieve file, as in this example:

```
35,00001This is a test record       YN
35,00002This is another test        NN
35,00003This is my last test record YY
```

The first few characters is an ASCII string containing the length of the record. This is a lot of overhead for fixed-length records, but it ensures compatibility with variable-length records as well.

After the record length, a comma is inserted into the file.

Following the comma, the entire binary image of the Btrieve record is placed in the file. The number of bytes in this image must correspond to the length specified for this line.

Finally, a CR/LF pair terminates each record.

Although it is possible to view and edit an unformatted file with a regular text editor, this is not recommended, as many editors can strip out important binary information from the data. Instead, use a binary or hex editor for making changes to these files, if you must. The Goldstar Software utility ViewUNF can be used to view the UNF files from a Windows interface with much less difficulty than seen in NotePad or WordPad.

CLONE
The CLONE operation takes an existing Btrieve file and creates a duplicate of it (in most respects) which contains no records and no preallocated space. For a clone to work successfully, the file header information must be intact. The format for this command is:
   BUTIL –CLONE <OutputBtrieveFile> <SourceBtrieveFile>

The new file will contain the same record size, page size, number of keys, etc., and is essentially a duplicate of the original file. In the Btrieve 6.x format, it is possible to drop any keys from the file, and it is possible that a BUTIL –CLONE process will renumber keys. As a result, you should proceed with caution if you know that keys have been altered in a file.

LOAD
The LOAD operation is the reverse of a RECOVER or SAVE operation. It reads an unformatted file and loads those records into a Btrieve data file. The format for this command is:
   BUTIL –LOAD <UnformattedFile> <BtrieveFile>

The data will be loaded into the file in the same order it was saved in the UNF file. Be sure to have sufficient disk space available to complete the load process, which will take some time. The resulting Btrieve file may actually be larger than the original, especially with newer files, due to the bundling of operations and shadow paging. If many records were deleted from the file, then the new file may be smaller than the original, too. In short, you should NEVER check the success of a rebuild by file size, but rather by doing a BUTIL -STAT of the file and checking the final record counts.

Versions of BUTIL
There are three different command-line versions of BUTIL: BUTIL.EXE for DOS, which runs on DOS machines with the local Btrieve engine or DOS requesters, BUTIL.EXE for the Windows NT Command Prompt, and BUTIL.NLM for NetWare servers. Additionally, there is a Windows GUI application called Btrieve File Manager (later called the Maintenance Utility) that supports some of the functionality.

DOS Version
The DOS BUTIL command is used primarily with Btrieve for DOS, but it was also a popular utility to use with Btrieve for NetWare for a long time. Skip this guy if you can, as he tends to be slow.

Windows Console Version
If you have a Windows database engine (either Server or Workgroup), or if you have any of the newer Pervasive.SQL clients, then you'll have a true Windows Console version of BUTIL. This software is not a DOS application, but is truly a 32-bit console application, and it uses DLL's to access the files via the client. Because it can run on the server itself, this is the preferred method for accessing files on Windows servers.

NetWare Console Version
Another version of BUTIL similar to the DOS version, this one runs from a NetWare server itself. Because the NetWare console is usually more "off limits" than other machines, many sites do not use BUTIL.NLM when they could benefit greatly. Instead, they revert back to the old DOS BUTIL.EXE. Unfortunately, they just don't realize that the NLM version, because it does not use the network for data transfer, runs 10-100 times faster depending on the data set.

Windows Applications
Depending on your version of Btrieve, you may have the 16-bit WBManage utility, the 32-bit Btrieve File Manager, or even the newer 32-bit Maintenance Utility. Regardless, each functions in a similar way -- it provides a GUI front-end to the maintenance steps of BUTIL, and each is both more and less capable than normal BUTIL. These tools allow you to recover files through menu-based commands, but there is no way to automate the process. The functionality of this tool is identical to BUTIL, but you should be aware that the "RECOVER" functionality is hidden under "Data/Save" menu option, and the CLONE feature is actually implemented with the "Show Information Editor..." option by loading the information from the original file and then creating a new file based on that information.

Using BUTIL to Rebuild Data Files
It was no coincidence that the most common BUTIL functions are STAT, RECOVER, CLONE, and LOAD. These are the exact commands required for rebuilding Btrieve data files to recover from corruption or to regain space from deleted records.
1. Before starting the rebuild, use BUTIL –STAT to print out the file details. Note especially if there are variable-length records or file compression, which will make recovery much more difficult. Save this printout for later use.
2. Next, make a backup copy of the file. If the rebuild fails of something else goes wrong, you may need this backup copy.
3. Verify that you have real corruption, and not phantom corruption. Return to the section above for details on the phantom corruption for the various engines.

4. From the server console, do a BUTIL -RECOVER on the file to create the unformatted file. Optionally, if enough of the file is damaged, you may wish to use DataSave, or another recovery utility of your choice to create this file.
5. Next, recreate the data file. If you have a program that creates the data files, use it to ensure that the file is correct. If you have a Btrieve Description file, use BUTIL -CREATE to make the new file. Otherwise, you may have to resort to a BUTIL -CLONE operation.
   Note that if the file header information is damaged, a BUTIL -CLONE may not create a proper data file. In case, you may wish to contact the software developer for a new file, or you can restore a good file from backup and clone that one.
6. The next step is to reload the data from the unformatted file into the new data file with a BUTIL -LOAD operation. This is usually the part that takes the longest amount of time, because the indexing information is continually being rebuilt at the same time.
7. Print out a BUTIL -STAT report for the resulting file and compare it to the original. If you are having a good day, you should see that all records, or perhaps all but a few, were recovered and loaded into the new file. In general, if a page is damaged, you will lose the group of records on that page Unfortunately, there's no easy way to determine which records were lost, outside of comparing the resulting file to the old backup, record by record.
8. Finally, clean up after yourself. Archive the backup file (and, optionally, the UNF file) to tape or CDROM and delete it. If you are using NetWare, run a PURGE on the directory to ensure that all disk space is properly freed up.

BREBUILD

While not officially a data corruption recovery tool, BREBUILD (RBLDCLI, or the Rebuild Utility on Pervasive.SQL V8.5 or newer) allows you to convert a file from one file format to another, and make minor changes (such as page size) in the process. It is critical to note that this should NEVER be used on a damaged file, as it lacks the error recovery features needed to fix problems in the file. Instead, the Rebuild tool simply stops when it encounters an error.

This process can be very lengthy (the time often increases **exponentially** with the size of the file). For example, on a server with lots of memory, a 150MB file may complete in 3-4 minutes. However, on the same server, a 768MB file with the same number of keys may take over 2 hours!

There are only two known reasons why you would NOT want to convert files to a newer format. First, you need to replicate the data (or portions thereof) to a server workstation that is still running an older engine. While newer engines can read older files, the reverse is not true. The other reason is that your application specifically creates file which are not supported in the newer version. (However, many newer engines will make the changes automatically, so that the app doesn't fail.) You could still convert all other files, however, and if the application does not need to create the file itself, you could even rebuild the file to a new page size to avoid the problem.

Usage of BREBUILD or RBLDCLI is easy, as it runs on either the server or on a workstation. You specify the file(s) to rebuild, and it does its thing. Using the -C parameter automatically skips files which are NOT Btrieve files, and this switch allows you to easily use wildcards when specifying the files to convert.

Optionally, you can even convert all page sizes to a constant value, or even have BREBUILD calculate an optimum size for each file for you. One note: BREBUILD from Btrieve 6.15 has been known to fail on later NetWare versions, and may be related to recent updates to CLIB. However, the BREBUILD from Btrieve 6.10 can be used prior to installing Btrieve 6.15 on a server. We have also seen problems with RBLDCLI on NetWare returning Status 170 or 171, but this can be avoided by providing a database login name manually when entering the command.

The Windows (GUI) version of BREBUILD, called the Rebuild Utility, has many of the same features, but it doesn't support wildcards for filenames. This makes it difficult to update mass quantities of files quickly. (When we updated over 26,000 files from 5.x to 6.x for a client, the GUI version would have been a nightmare!)

DataSave – Jim Kyle to the Rescue

Originally called DUMPDATA, this utility from Jim Kyle, the author of *Btrieve Complete*, is a must-have. It allows you to recover data from a Btrieve 6.x file into an unformatted file **without even having Btrieve loaded**! Assuming that most of the data pages are intact, this program can successfully recover a majority of the data in 90% of all corruption cases.

It works by reading the data file, manually interpreting what it finds there and trying to sniff out the proper records. It does an excellent job for just about every file with fixed-length records. Depending on the corruption, problems may surface for variable-length or compressed record files, but these are the exception, not the rule. DataSave cannot handle 7.x or newer file format,s but it does offer an interesting option for older databases.

You can contact Jim Kyle at jimkyle@acm.org or visit www.jimkyle.com for more information.

PATFIX

PATFIX is a Goldstar Software utility (www.goldstarsoftware.com/tools.asp) that helps recover from corrupted Page Allocation Tables. It was created to specifically address a corruption issue in the 32-Bit Btrieve 6.15 for Windows 95/NT engine, but may have other uses as well.

Versions of the 32-bit Btrieve engine before 6.15.451 has a problem in that the FCR and PAT pages are not properly updated. With the use of Shadow Paging, the engine should alternate the two copies of these internal pages. Unfortunately, in this specific engine, the compiler used on the engine created bad code, and only one copy is in use. In the event of a workstation engine failure, the engine tries to roll back the file to its previous

state by setting the usage counter on the active PAT pages to 0. Unfortunately, since the other pages were never updated, this effectively removes all records from the file, causing massive data loss.

This corruption usually surfaces after an engine crash, and is indicated by a BUTIL - RECOVER recovering either 0 records, or only a small fraction of records. (In essence, one page of data is the most that *can* be recovered.) Due to the nature of the corruption, even DATASAVE fails to recover any data, and often crashes the workstation it is running on due to pointer errors.

The PATFIX program displays the two copies of the FCR and each set of PAT's and allows the user to select the page that should be active. (The 32-bit version supports auto-fixing as well.) After running PATFIX on a file, keep in mind that the last updates may have been incomplete, and the internal structures may be damaged. Immediately make a backup and follow the standard RECOVER/LOAD process on the file before using it again.

What If ALL Recovery Methods Fail?

If these utilities are unable to recover all of the data, your best bet may be to bite the bullet and restore the entire database from your most recent backup. Yes, the ENTIRE database. You may be able to restore just the corrupted file, but many databases often contain inter-linked information that must remain consistent between files. Only the application developer can tell you for sure what the proper process is for restoring in this case. As such, you should always consult with the developer before a single-file restore, unless you can restore all files back to a known-good point. (If you don't have good backups, it's probably too late right now, but we recommend checking out our white paper on proper database backups.)

If the data is vitally important and you have no way to restore the missing information, there are people available who can provide recovery assistance for a fee, including Jim Kyle, Classic Software, and Goldstar Software. The cost for such a manual recovery is usually a lot less than losing lots more data, but it is a lot more expensive than simply restoring from backup.

For the do-it-yourselfers of the world, the best recommendation is a copy of *Btrieve Complete*, a good hex editor and file viewer, a pad of paper, and a lot of patience. The typical manual recovery process can run anywhere from 1 to 6 hours, depending on the size of the file and the amount of the corruption.

The Goldstar Software has also created some important tools that can help aid the corrupt file analysis process. The Goldstar Software File Recovery Toolkit contains a number of very useful tools for database file recovery, including the automatic version of PATFIX (mentioned above), and other tools such as PATCheck, GSRecover, ListDPg, and more. For information on this toolkit, which has been recently updated to help recover data from 8.x and 9.x files as well, check out website at www.goldstarsoftware.com/pdfrt.asp.

If you are not one that thinks digging through hex dumps is a good time, then let us help you through this process! Goldstar Software also offers a full range of data file recovery services to analyze your files and recover as much data as possible when the worst occurs. Get more information from http://www.goldstarsoftware.com/recovery.asp.

And, to help prevent the loss of sleep that often accompanies data loss, allow our database experts to work with you to create a multi-tiered backup system to protect your data via our Database Protection Services. Get more information from http://www.goldstarsoftware.com/gsdps.asp.