

PATCheck v5.55 for Win32

Program Description

The PATCheck application opens the Btrieve database file as a standard binary file, looping over all valid PAT Pairs. For each PAT pair, it reads the Logical Page numbers within that PAT, looks up the corresponding physical location for each logical page, and verifies the logical page signature located at that physical location.

While PATCheck only validates the integrity of the PAT pages themselves, it can be used to review the basic integrity of a file more quickly than KeyCheck.

The Win32 version of the PATCheck utility is much more robust and faster than the DOS version. The DOS version should be fully retired.

This version of PATCheck also includes the capability to fix up the most common errors, including damaged PAT headers and bad pointers. Including an option of /F255 will fix all found problems (if possible). The "8" flag is the most thorough flag available, as it attempts to locate a missing page when it finds a bad pointer. As such, this option can drastically increase the processing time.

The /F option is NOT available in the Evaluation license.

Platform and Package

Win32 Console; Btrieve Advanced Recovery ToolKit; GSLic

Pricing

Available as part of the BART ToolKit Only.

Command Line Syntax and Help Screen

PATCheck Version 5.55: 01/14 (C)2026 Goldstar Software Inc.
Using Low-Level File I/O Version 2.42.15

Usage: PATCHECK SrcFile [/F#] [/R] [/O] [/S#] [/U] [/Z]

Scans a file's Page Allocation Table and report any inconsistencies.
Use the /O Flag to display PAT and physical page Offsets.
Use the /R Flag to suppress progress reporting every new PAT pair.
Use the /S# Option to start processing at PAT Pair #.
Use the /F Option to fix the specified problems. (Registered Only)
Add up these options (or OR the values) to fix multiple issues.
1 = If PAT points to Physical Page with 0x00000000 PageID, fix PageID.
2 = If PAT points to Invalid Physical Page, make it free space pointer.
4 = If any PAT Header data is Damaged, fix it.
8 = If PAT points to the wrong page, scan to find the best page & fix it.
Use /U to open files in UNPROTECTED mode. (Allows concurrent access.)
Use /Z to check PAT pointers with a page ID of zero.
WARNING! Use of this program without a proper BACKUP is prohibited.

Examples and Sample Usage

The following text is an example run on a file with a corrupted PAT entry:

PATCHECK Version 5.54: 07/11 (C)2024 Goldstar Software Inc.

Scanning file CORRUPT.DAT for PAT inconsistencies...

Checking PAT Pair 1...
LogPAT 0x00000001(44) -> PhysPage 0x00000020 with LogPage 0x00aaaaaa(aa).
LogPAT 0x00000002(80) -> PhysPage 0x0000003a with LogPage 0x00aaaaaa(aa).
LogPAT 0x00000004(44) -> PhysPage 0x00000010 with LogPage 0x00aaaaaa(aa).
LogPAT 0x00000006(44) -> PhysPage 0x00000012 with LogPage 0x00aaaaaa(aa).
LogPAT 0x00000009(44) -> PhysPage 0x00000039 with LogPage 0x00aaaaaa(aa).
Scan complete.
Total Problems Found: 5

In this example, the PAT indicates that logical page 1 is a Data page (page type 0x44) at physical page 0x00000020. However, when the page at the corresponding physical location (0x20 * page size) was read, it had an invalid page header.

Add the /O option to see the physical offset into the file of each PAT pair and physical page. This can aid in viewing the point of corruption manually.

Use the /R option to eliminate the periodic status reporting of the PAT page that the program is currently processing. (This is useful for generating reports about a specific data file's corruption.)

Use the /S# option to specify starting at a given PAT Pair Number. This option may be useful for debugging purposes, and should not typically be used.

Adding the /U flag will open the files in unprotected (shared) mode, so that you can look up data in files that are open by other applications, such as the hex editors or other tools. Note that accessing files in this mode that are active by the PSQL engine is considered a bad thing. Also, you cannot use the /U switch with the /F switch.

Normally, PAT pointers with a zero page ID can be safely ignored. PATCheck will report a count of these issues, but will not display data for them by default. In some cases, you may wish to display (and fix) these pointers, too, so use the /Z switch to handle zero page ID's.

Licensed users of PATCheck can include the /F option and indicate the types of problems to fix. You can do multiple fixes at the same time by adding these values together (they are bit flags) and using that number. For example, /F12 will do both option 4 and option 8 at the same time.

- Option 1: If the PAT points to a page, but that page is found to have a zero (i.e. invalid) Page ID, fix it up by writing the proper page ID into the target page. (This is only valid when the PAT is valid, but pages have been blanked due to a hardware failure.)
- Option 2: If the PAT points to any page which is outside of the file itself (i.e. beyond EOF) or otherwise does not have a valid page ID, change the PAT by changing the page pointer to an empty page. The net result of this is that the page will be removed from the file completely, and data loss will result if the pointer was to a data page. This option is the only viable option right now for page-compressed files.
- Option 4: Analyze the PAT header data itself, report any issues, and attempt to fix them up if possible. This may not always make good choices, so if you have a file that gets worse when this flag is used, send us your sample data.
- Option 8: The slowest option, but perhaps the most useful, is /F8. This causes PATCheck to detect bad pointers, and then search through the list of actual page ID values in the file to find the "best" copy of that page (based on the usage count of the copies that it is able to find). For regular files, PATCheck will pre-read all header values for optimal speed. This option is NOT viable for page-compressed files, as the usage count is embedded inside the compressed data block and not stored with the header itself.

Here is an example of a real PATCheck run with the /F8 option:

```
PATCheck Version 5.54: 07/11 (C)2024 Goldstar Software Inc.
I sure hope you had a backup....
Scanning file badpat.dat for PAT inconsistencies...

Checking 0x00000474 pages...
Scanning PAT Pairs from 1 through 2...
Checking PAT Pair      1...
LogPAT 0x00000118(2020) -> PhysPage 0x0001011a (beyond EOF or NULL PageID).
Found Logical Page 00000118 at Physical Page 0000011a with type 0044.
```

```
-->Repaired By Changing Page Pointer to (0044)0000011a.
LogPAT 0x00000119(2020) -> PhysPage 0x20202020 (beyond EOF or NULL PageID).
Found Logical Page 00000119 at Physical Page 000000b8 with type 0044.
-->Repaired By Changing Page Pointer to (0044)000000b8.
LogPAT 0x0000011a(4c46) -> PhysPage 0x4b202020 (beyond EOF or NULL PageID).
Found Logical Page 0000011a at Physical Page 00000111 with type 8000.
-->Repaired By Changing Page Pointer to (8000)00000111.
LogPAT 0x0000011f(0000) -> PhysPage 0x00000000 with LogPage 0x00004346(0043).
Found Logical Page 0000011f at Physical Page 0000008e with type 0044.
-->Repaired By Changing Page Pointer to (0044)0000008e.
```

<lines removed>

```
LogPAT 0x000001ad(ffff) -> PhysPage 0xffffffff (beyond EOF or NULL PageID).
Found Logical Page 000001ad at Physical Page 000001cc with type 0044.
-->Repaired By Changing Page Pointer to (0044)000001cc.
Checking PAT Pair      2...
Scan complete.
```

```
Problems Found (PAT Pages)   : 0
Problems Found (Data Pages)  : 118
Problems Found (Index Pages) : 32
Problems Found (Other Pages) : 0
Problems Found (Total)      : 150
Problems Fixed (Total)      : 150
```

```
PAT Headers Fixed:          : 0
Data Page Pointers Fixed    : 118
Index Page Pointers Fixed    : 32
Variable Page Pointers Fixed: 0
Pages Reset To Free Space    : 0
Problems Fixed (Total)      : 150 (May not be sum of above)
```

Other Information

PATCheck is part of the Btrieve Advanced Recovery Toolkit (BART).

For more information on these utilities contact us at www.goldstarsoftware.com

Version History

Version 2.0: First documented version

Version 2.1: Fixed problems with missing physical pages beyond EOF.

Version 2.2: Fixed bugs with 7.x extended files (tested to 5GB).

Version 2.4: Reduced line length of error messages

Version 3.0: Rewritten to use low-level I/O Routines, Supports V8 files.

Version 5.0: Major Revision: Updated Low-Level IO for PSQlv9.5 File Formats. Improved reporting detail on fixed pages. Added ability to fix PAT Pointers that pointed beyond EOF or to a NULL page.

Version 5.1: Fixed bugs in handling v9.5 PAT pages.

Version 5.2: Added GSLic capability.

Version 5.30: Fixed issue with PATCheck not properly fixing all PAT entries when the PAT had actually been overwritten with garbage. Changed BestPage search algorithm to accept index pages with a usage counter of 0.

Version 5.40: Added ability to pre-load all page headers into memory before starting the process and access all data directly from RAM during the check. This resulted in a *substantial* performance gain for large files.

Version 5.41: Improved error reporting, with totals of the fixes that were actually made. This should make it easier to see how effective a fix might have been.

Version 5.42: Updated licensing code.

Version 5.43: Added status reporting for page header pre-load.

Version 5.44: Fixed error in status reporting on small files.

Version 5.45: Added ability to display PAT/page offsets.

Version 5.46: Added ability to report (but not fix) bad Data Page Usage Counts.

Version 5.47: Added Unprotected Mode option.

Version 5.50: Added support for v13 file format.

Version 5.51: Split out the PAT problems into a new counter, and treat pages with a zero Page ID as “soft” errors, requiring /Z to scan and fix.

Version 5.52: Found a way to validate PATs for page-compressed files. Currently, Fix Option 2 is the only viable way to fix it, which blanks all invalid page pointers.

Version 5.53: Changed validation check for Usage Count to validate only when the 'TARGET' page is a data page to reduce false error count.

Version 5.54: Improved support for v13/v16 file formats.

Version 5.55: Added support to properly detect DE pages in the PAT that link to D pages in the file and avoid flagging these as errors.

Known Problems

Files with page compression enabled are only minimally-supported, and encrypted files are not supported at this time.