

Memory File Performance Pack

Making optimal use of your system memory

Key Features

- Improved performance
- Easy to use
- Reports on memory savings
- Transparent to your PROIV applications

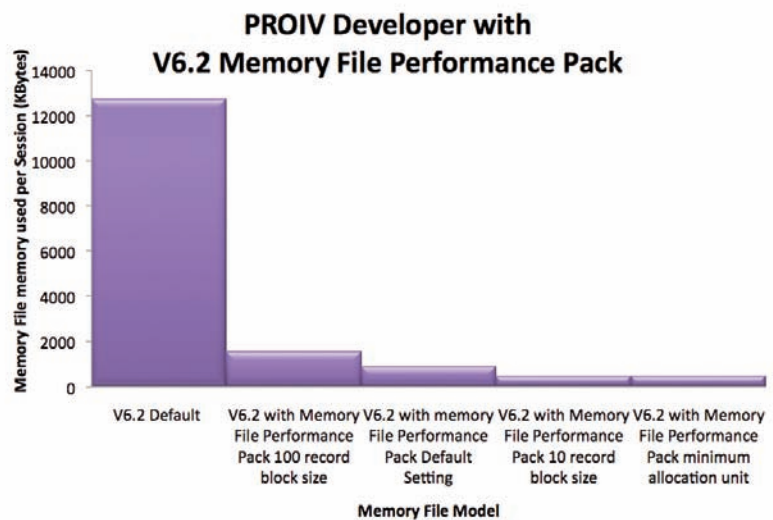
Benefits

- Optimises use of system memory
- Available on all supported PROIV platforms
- Defer need for expensive hardware upgrades

Memory files were introduced in PROIV Version 4.6 as a mechanism for storing record data without the need to write it to physical storage. The record data was held in memory and not shared between PROIV sessions and as a result was extremely fast.

Memory files are directly analogous to PROISAM files in their operation. Using them gives the benefit of bypassing the I/O processing and concurrency requirements of a physical file thereby giving a higher level of performance. However, this gain is not without cost. Memory files require a significant amount of memory to operate; excessive use of them can significantly increase the memory requirements of a PROIV application.

As part of NorthgateArinso's drive to improve performance and reduce resource requirements of the core PROIV Virtual Machine, memory files have been updated to make better use of system resource. Improvements to the file structure allocation model mean that now you have better control over the memory allocated by the memory file interface.



The details

The benefits to the application server are clear. There is a potential to save a significant amount of system memory - clearly the more you make use of memory files the more memory you could save. In the example on page one of this datasheet, PROIV Developer makes use of approximately 25 memory files and the savings are approximately 90%. Memory is a significant saving to an operating system; processes requiring less memory allow the operating system to work more efficiently thereby resulting in faster application execution. On a 32-bit Windows system this could be quite beneficial given the 2Gb process size limit.

How do I know if my application will benefit from the Memory File Performance Pack

If your application uses memory files then it almost certainly will; the real question is how much benefit will you get? To help you understand this better an improved tracing mechanism has been added to the memory file subsystem. The tracing allows you to experiment with different memory file configurations and to see their effect on memory utilisation when you run parts of your application.

As an example below the following snippets of trace file show the effect of varying the size of the memory allocation unit on the overall occupancy of the storage for memory files within PROIV Developer.

The details

The PROIV Version 6.2 Memory File Performance Pack is a significant enhancement to the existing memory file subsystem; the more you use memory files the bigger improvements in resource utilisation you will see from your application. Gains seen could be as much as a 90% reduction in memory used by your memory files. To recognise this, the Memory File Performance Pack will be a standard component of PROIV Developer and will be available in the November 2009 release of PROIV Version 6.2.

This exciting new product option is also available to trial within your PROIV Runtime systems; contact your account manager for further details.

Version 6.2 Default

```

...INFO ... Filetype: MEMORY, Name: @vd011
...INFO ... Record key size: 32 bytes (plus 20 byte overhead)
...INFO ... Record data size: 221 bytes (plus 3 byte overhead)
...INFO ... Record stats: Peak records used: 129, allocated: 1000
...INFO ... Peak space used: 35604 bytes, allocated: 276000 bytes. [12% occupancy]
...INFO ... [1 block(s) allocated @ 1000 records each, plus 8 byte block overhead]
        
```

Version 6.2 with Memory File Performance Pack

```

...INFO ... Filetype: MEMORY, Name: @vd011
...INFO ... Record key size: 32 bytes (plus 20 byte overhead)
...INFO ... Record data size: 221 bytes (plus 3 byte overhead)
...INFO ... Record stats: Peak records used: 129, allocated: 150
...INFO ... Peak space used: 35604 bytes, allocated: 41400 bytes. [86% occupancy]
...INFO ... [3 block(s) allocated @ 50 records each, plus 8 byte block overhead]
        
```