Why can’t I use SEC_LARGE_PAGES with a file-based file mapping?

devblogs.microsoft.com/oldnewthing/20250409-00/?p=111061

April 9, 2025

A customer wanted to create a memory-mapped file with large pages. They took the sample code and adapted it by changing

```
hMapFile = CreateFileMapping(
     INVALID_HANDLE_VALUE,    // use paging file
     NULL,                    // default security
     PAGE_READWRITE | SEC_COMMIT | SEC_LARGE_PAGES,
     0,                       // max. object size
     size,                    // buffer size
     szName);                 // name of mapping object
```

to

```
hMapFile = CreateFileMapping(
     hFile,                   // use a specific file
     NULL,                    // default security
     PAGE_READWRITE | SEC_COMMIT | SEC_LARGE_PAGES,
     0,                       // max. object size
     size,                    // buffer size
     szName);                 // name of mapping object
```

However, this failed with `ERROR_INVALID_PARAMETER`.

The reason for the failure is documented: The `SEC_LARGE_PAGES` flag cannot be used with file-based file mappings. It can be used only with pagefile-based file mappings.

But why?

Recall that on Windows, large pages are not pageable. A non-pageable mapping backed by the pagefile is basically memory that can never be paged out. The “backed by the pagefile” is just an accounting artifact; the memory never actually goes to the pagefile since it never pages out.

Although you can ensure that large pages that are file-backed never get paged out, you still have another problem: Mappings that are backed by a file still need to be flushed when the mapping closes. This means that you have a large page’s worth of I/O (or more

likely, multiples of them) all pending when the handle closes, and that's not something the memory manager has bothered implementing.

Recall that the original audience for large pages is high performance computing scenarios where the entire system is dedicated to running a single program like SQL Server, and those programs don't want paging to happen in the first place. Either the entire workload fits into memory, or you need more memory. (Related.) There's no point implementing a feature (large pages in memory-mapped files) that nobody has asked for.

But maybe this customer wants to ask for it. If they could explain why they needed large pages for file-based mappings, that could help tip the scales toward convincing the memory manager team to implement the feature.