Automated Memory Dumps Using Just-in-Time/Postmortem Debugging

- Microsoft offers several debugging tools to capture application crash dumps, for example CDB and NTSD[1].
- We will use NTSD to capture memory dumps of real time application crashes.
- Install the x64 and x86 versions of Debugging Tools for Windows.
- Configure Windows 11 to create unique dump files for both x86 and x64 bit applications.
- Configure Windows Error Reporting Services:
    - Set HKLM\Software\Microsoft\Windows\Windows Error Reporting\LocalDumps\DumpFolder to C:\Coredumps. It's of type REG_EXPAND_SZ.
    - Set HKLM\Software\Microsoft\Windows\Windows Error Reporting\LocalDumps\DumpType to 2 which means Full dump. It's of type REG_DWORD.
    - Disable UI prompts for crash dumps by setting HKLM\Software\Microsoft\Windows\Windows Error Reporting\DontShowUI to 1. It's of type REG_DWORD.
- Install the x86 as well as x64 versions of Debugging Tools for Windows to their default locations. The x86 version will be used to capture dumps for x86 applications and x64 version for x64 applications.
- Create a directory C:\Coredumps.

---

[1] https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/enabling-postmortem-debugging

- Enable AeDebug by running following commands:
    - “C:\ Program Files (x86)\Windows Kits\10\Debuggers**x86**\ntsd.exe” -iaec “-c \”.logopen /t c:\coredumps\JustInTime.log; !analyze -v; ~*k250; lmv; .dump /mA /u c:\coredumps\JustInTime.dmp;q\””
    - “C:\ Program Files (x86)\Windows Kits\10\Debuggers**x64**\ntsd.exe” -iaec “-c \”.logopen /t c:\coredumps\JustInTime.log; !analyze -v; ~*k250; lmv; .dump /mA /u c:\coredumps\JustInTime.dmp;q\””
- More information about the flags used below.
- Run the crashtest.exe to check if NTSD kicks in and captures memory dumps in c:\coredumps\. Source code below.
- Check following registry keys every time you reboot the machine to ensure NTSD is the active debugger. I’ve seen it switching to VSDebugger frequently.
    - HKLM\Software\Microsoft\Windows NT\CurrentVersion\AeDebug\Debugger
    - HKLM\Software\WoW6432Node\Microsoft\Windows NT\CurrentVersion\AeDebug\Debugger
    - Make sure these are set to the command we ran earlier.
- Modify the following registry key to turn off the pop up.
    - Set HKLM\System\CurrentControlSet\Control\Windows\ErrorMode to 2 which always logs error messages to Windows event logs and responds OK to hard errors.
- Configure Windows to download symbols (PDB files) by creating an environment variable _NT_SYMBOL_PATH and setting it to Microsoft symbol server.
    - Srv*C:\Symbols*https://msdl.microsoft.com/download/symbols

Appendix

Crashtest.exe source code

```
#include <iostream>


int main() {
    int* ptr = nullptr;
    *ptr = 5; // Dereference a null pointer to intentionally crash
    return 0;
}
```

NTSD flags used.

-i: This option specifies that NTSD should run in initial breakpoint mode. When a process starts, NTSD will break execution at the entry point of the process.

-a: This option specifies that NTSD should automatically attach to a process specified by its process ID (PID). This is useful for attaching the debugger to a running process without user intervention.

-e: This option specifies that NTSD should create a new process for the target being debugged. This is useful when you want to debug a new instance of an executable rather than attaching to an existing process.

-c: This option specifies a command to execute when the debugger attaches to the target process. This allows you to automate certain debugging tasks or execute specific commands upon attachment.

/mA: This option specifies a "Full" memory dump. A full memory dump contains the entire contents of physical memory (RAM) at the time of the crash. This includes kernel-mode memory, user-mode memory, and any other memory allocated by processes. Full memory dumps can be very large, often several gigabytes in size, depending on the amount of physical memory installed on the system.

/ma: This option specifies an "Active" memory dump. An active memory dump only includes the memory that is actively in use by the operating system and kernel-mode drivers at the time of the crash. It excludes unused portions of physical memory, which can significantly reduce the size of the dump file compared to a full memory dump. Active memory dumps are generally smaller in size and may be more suitable for systems with limited storage capacity.

/u: This option includes user-mode memory in the memory dump. When used in combination with other options such as /mA or /ma, /u ensures that user-mode memory regions are also included in the dump file. This can be useful for analyzing crashes that involve user-mode processes and applications.

Including user-mode memory in the memory dump provides a more comprehensive view of the system state at the time of the crash, which can be helpful for debugging purposes.

So, when you use /u along with /mA or /ma, it ensures that both kernel-mode and user-mode memory are included in the memory dump file.

References:

https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/enabling-postmortem-debugging

https://community.progress.com/s/article/P148139

http://www.microsoft.com/whdc/devtools/debugging/installx86.mspx#a