Content Type	Support Article
Article #	000022747
Title	Capturing a Memory Dump File Using the Microsoft Debug Diagnostic Tool (64bit)
Legacy DocId	TN412
Confidence	Expert Reviewed
Published On	4/4/2022

Capturing a Memory Dump File Using the Microsoft Debug Diagnostic Tool (64bit)

Summary

Use the Debug Diagnostic Tool (DebugDiag) for troubleshooting issues such as hangs, slow performance, memory leaks or fragmentation, and crashes in any user-mode process. The tool includes additional debugging scripts for Internet Information Services (IIS) applications, web data access components, COM+ and related Microsoft technologies.

Note: This application supports 64-bit (x64) systems. For 32-bit procedure, see Capturing a Memory Dump File Using the Microsoft Debug Diagnostic Tool (32bit) .

This article provides instruction for creating a Memory Dump for a designated issue type.

Situation

Application Versions

- DebugDiag v2 Update 3

Generating Memory Dumps

To use DebugDiag effectively, you need to first identify what kind of issue you are troubleshooting (e.g. a crash, hang, slow performance, or memory and handle usage). This step will aid in configuring the tool appropriately to get the right data, then to identify the root cause of the problem in order to resolve it.

Preparing the Environment

1. Download the DebugDiag.msi file. This release only supports 64-bit systems.
2. Save the file to a folder of your choice and double-click it to begin the installation.
3. Open DebugDiag by clicking **All Programs / Debug Diagnostics Tool 2 / DebugDiag 2 Collection (Figure 1)**.

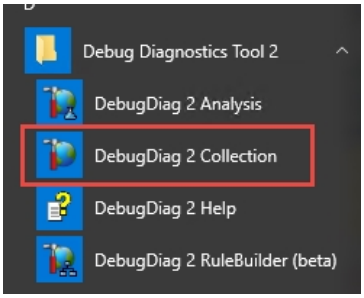

Figure 1: Open Debug Diagnostic

4. When the **Select Rule Type** window appears, click the **Crash** or **IIS Hang** option as appropriate, then click **Next** (Figure 2 below).

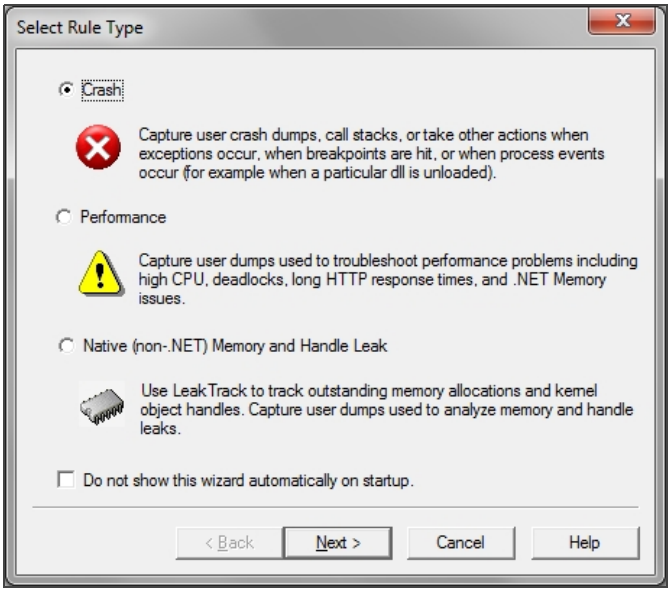

Figure 2: Select Rule Type

5. In the **Select Target Type** dialog box, click the **A specific process** option, then click **Next** (Figure 3 below).

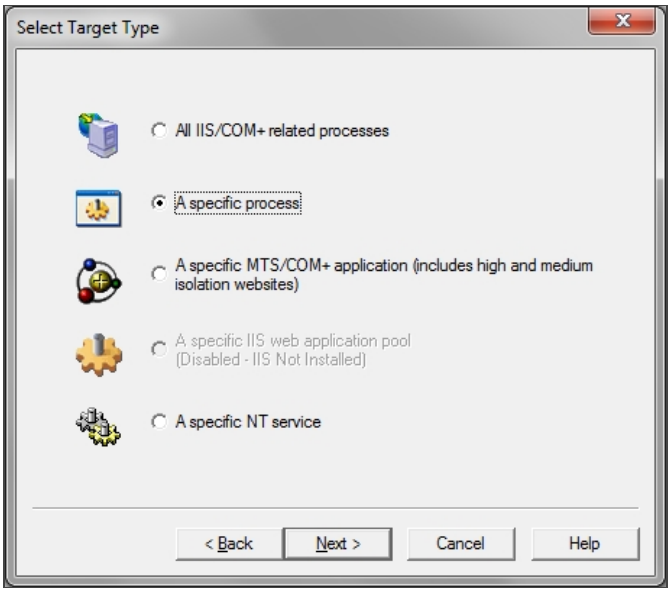

Figure 3: Select Target Type

6. Sort the target list by **Process Name** and find the process you need to troubleshoot. Click a process in the list to highlight it (Figure 4 below).
7. Make sure that the **This process instance only** option is NOT checked, then click **Next**.

Note: For this example, **view.exe** was crashing. However, the DebugDiag tool can be used for any process.

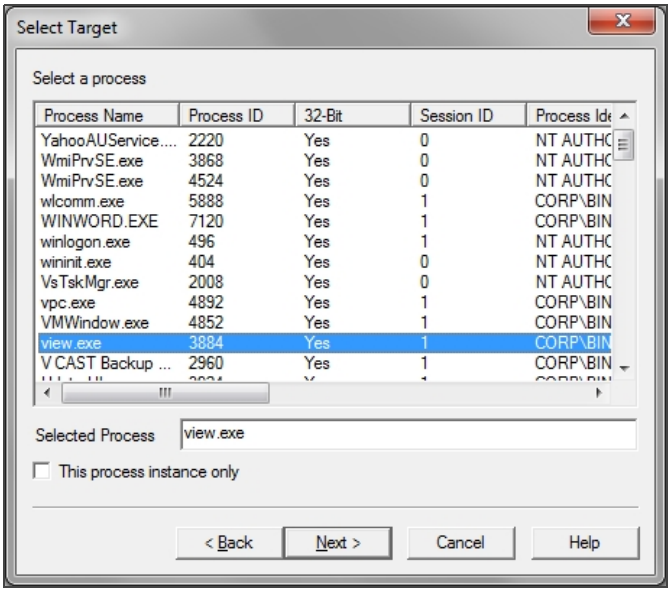

Figure 4: Select a Target Process

8. Advanced Configuration is optional, so if you do not need it click **Next** without configuring anything (Figure 5 below).

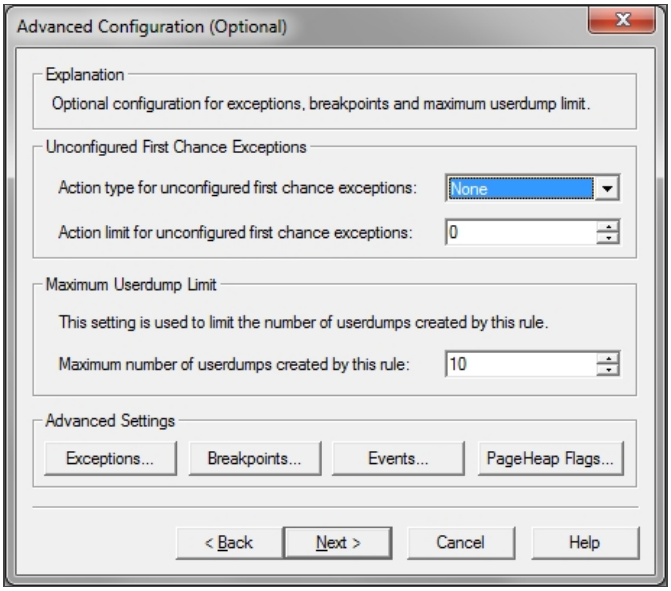

Figure 5: Advanced Configuration (Optional)

Optional Configuration Details

The following settings are optional but provide an extra level of control for configuring your dump file. For this example, you can set a few breakpoints and Exception codes using the **Advanced Configuration** dialog box.

If you want to skip Advanced Configuration details, go to Step 11 below.

1. Click **Exceptions** (Figure 5 above). The **Exception Configuration** window appears (Figure 6 below).
2. Click **Add Exception** (Figure 6 below).

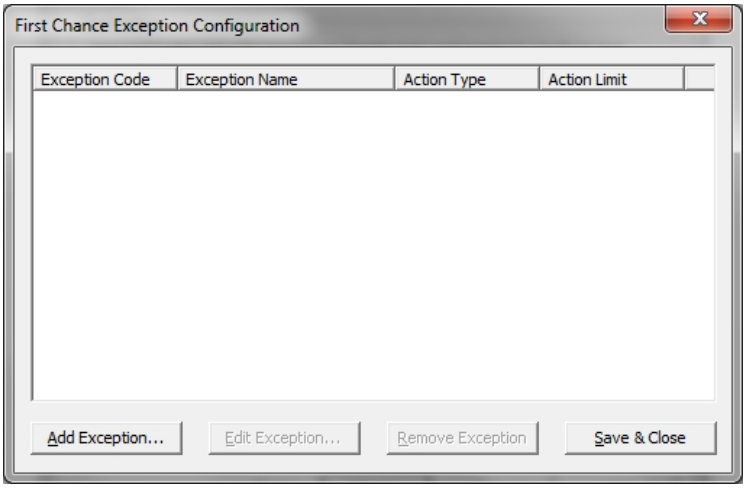

Figure 6: First Chance Exception Configuration

3. For this example, select the exception code **C0000005 Access Violation**.
4. Keep the **Action Type** set to **Log Stack Trace** and change the **Action Limit** to **0**. This setting indicates no limit on the number of callstacks to output to the log file whenever there is an access violation within the process.

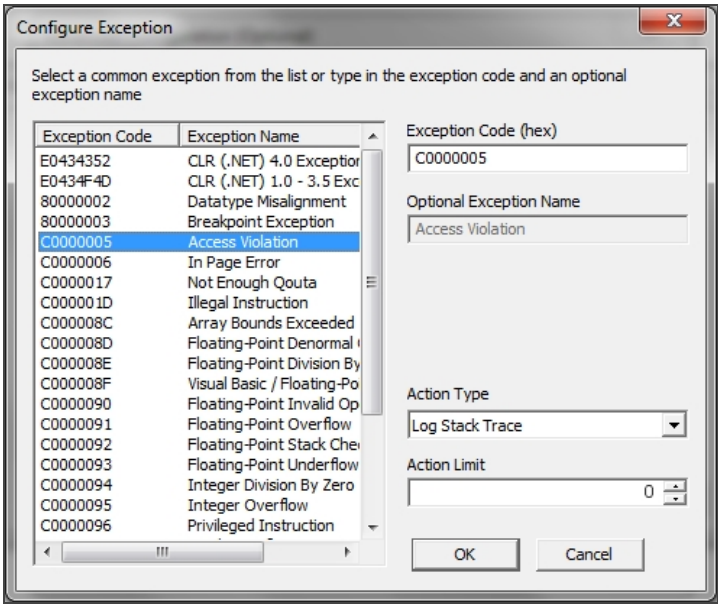

Figure 7: Select an Exception

5. Click **OK**. The Exception Code appears in the Configuration panel.

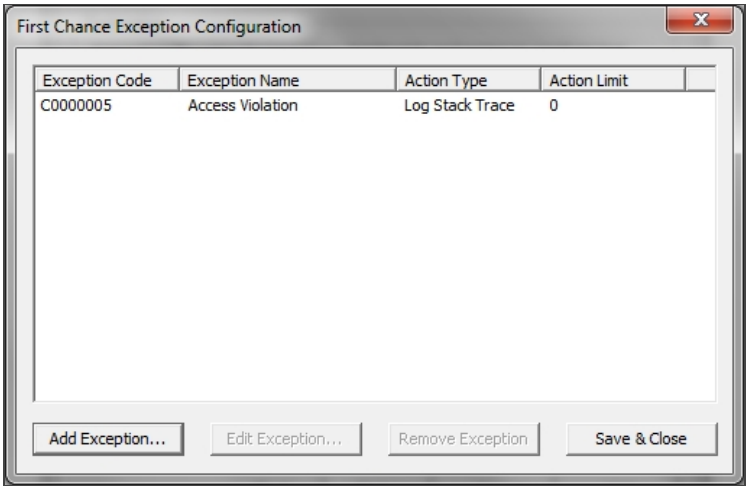

Figure 8: Exception Code Panel

6. Click **Save & Close**.
7. Click the **Breakpoints** button near the bottom of the dialog box (Figure 5 above). The **Configure Breakpoint** dialog box appears.
8. Click **Add Breakpoint** (Figure 9 below).

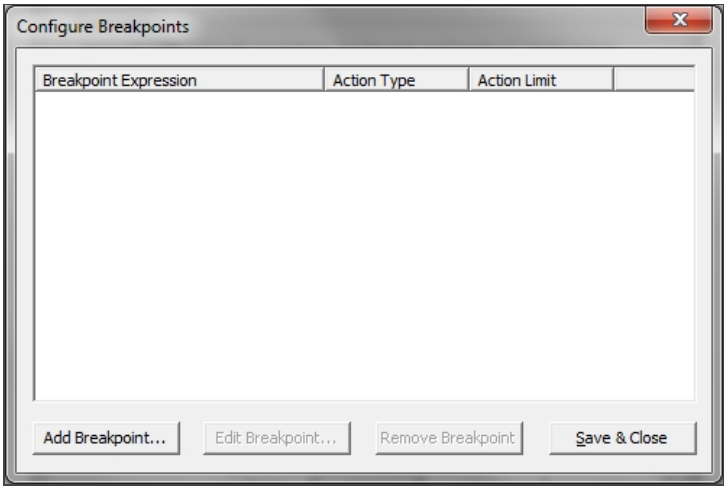

Figure 9: Configure Breakpoints Dialog Box

For example, if you select **mscorwks!AppDomain::OnUnhandledException** in the list, then select **Full Userdump** from the **Action Type** list and set an Action Limit of 1 (Figure 10 below).

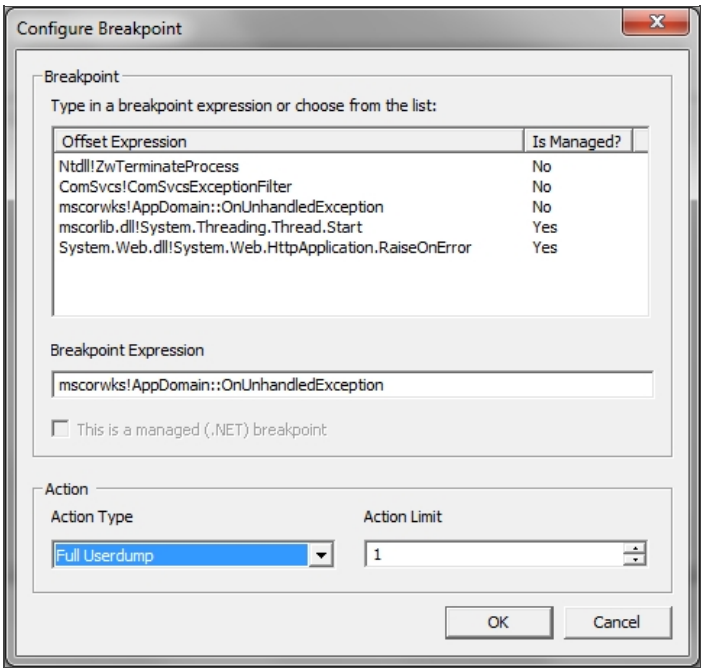

Figure 10: Configure Breakpoint

9. Click **OK**. The entry appears in the **Breakpoint Expression** panel (Figure 11 below).

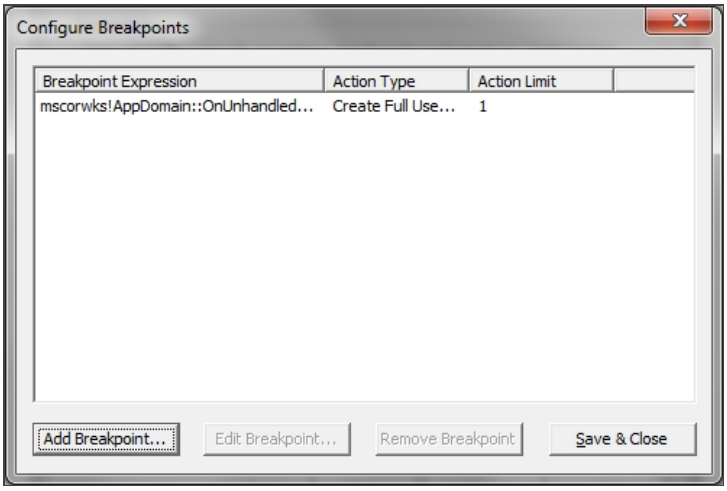

Figure 11: BreakPoint Expression

10. Click **Save & Close**.
11. On the **Advanced Configuration** dialog box click **Next**. (Figure 5 above).
12. Set the path for the dump file (Figure 12 below).

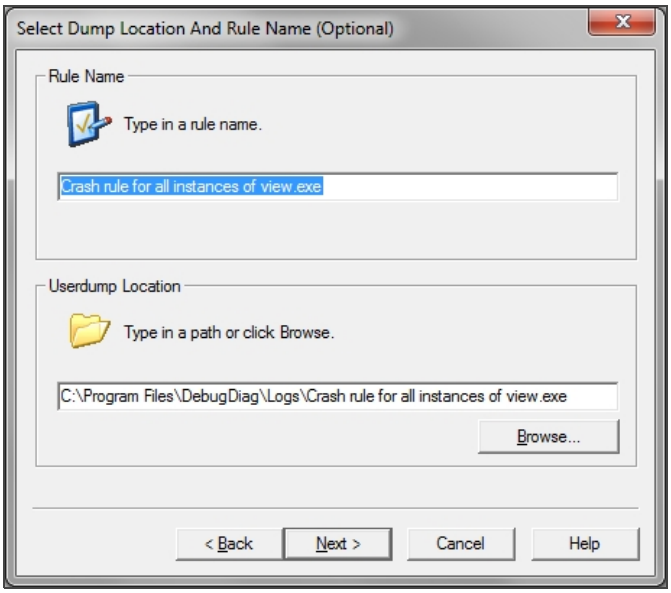

Figure 12: Set the Dump File Path

13. Click **Next**, then click **Activate the Rule** and **Finish** (Figure 13 below).

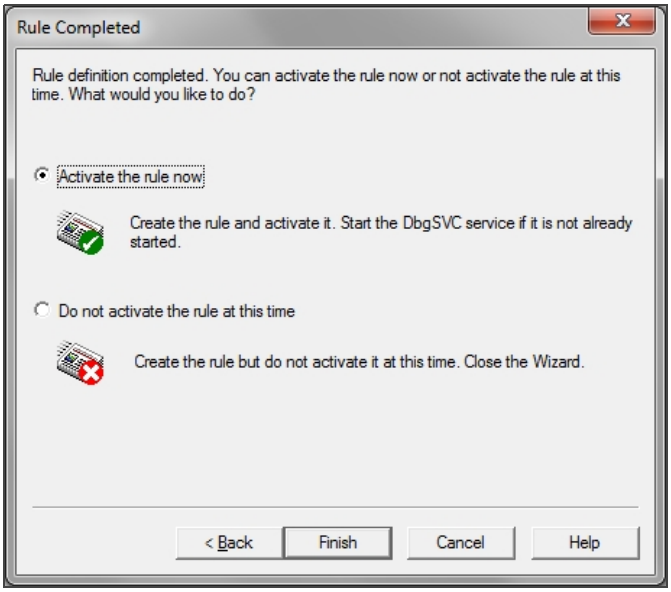

Figure 13: Activate the Rule and Finish Configuration

You now have attached DebugDiag to the process. If there are any access violations in this process you get a callstack of that failure in the log file that is created in that event. If the process shuts down for any reason you will get a full user dump of that process.

You can close the DebugDiag window since the debugger uses a service to help with the debugging. When you open DebugDiag again it displays a list of rules that are running and how many dumps it has collected thus far.

Review the Help Files for the DebugDiag Tool. They are very helpful and include lots of screenshots to illustrate. The help file is installed at: **C:\Program Files\DebugDiag\DebugDiag.chm**.