Using Dump Files

Visual Studio 2015
Other VersionsOther VersionsMicrosoft \square \square

Dump files with or without heaps; create a dump file; open a dump file; find the binaries, pdbs, and source file for a dump file.

Contents

What is a dump file?

Dump files, with or without heaps

Requirements and limitations

Create a dump file

Open a dump file

Find binaries, symbol (.pdb) files, and source files

What is a dump file?

A *dump file* is a snapshot of an app at the point in time the dump is taken. It shows what process was executing and what modules were loaded. If the dump was saved with heap information, the dump file contains a snapshot of what was in the app's memory at that point in time. Opening a dump file with a heap in Visual Studio is like stopping at a breakpoint in a debug session. Although you cannot continue execution, you can examine the stacks, threads, and variable values of the app at the time the dump occurred.

Dumps are primarily used for debugging issues that occur on machines that the developer doesn't have access to. For example, you can use a dump file from a customer's machine when you can't reproduce the customer's crash or hang on your machine. Dumps are also created by testers to save crash or hang data so that the test machine can be used for more testing. The Visual Studio debugger can save dump files for managed or native code. The debugger can load dump files that were created by Visual Studio or by other programs that save files in the *minidump* format.

• Contents

Dump files, with or without heaps

You can create dump files with or without heap information.

- Dump files with heaps contain a snapshot of the app's memory. This includes the values of variables at the time the dump was created. If you load a dump file that was saved with a heap, Visual Studio can load the symbols even if the application binary is not found. Visual Studio also saves the binaries of loaded native modules in the dump file, which can make debugging much easier.
- Dump files without heaps are much smaller than dumps with heap information. However, the debugger has to load the app binaries to find the symbol information. The binaries must be an exact match of the binaries that were used when the dump was created. Only the values of stack variables are saved in dump files without heap data.
- Contents

Requirements and limitations

- Debugging dump files of optimized code can be confusing. For example, compiler inlining of functions can result in unexpected call stacks and other optimizations might change the lifetime of variables.
- Dump files from 64-bit machines must be debugged on an instance of Visual Studio that is running on a 64-bit computer.
- In versions of Visual Studio before VS 2013, dumps of 32-bit apps that were run on 64-bit machines that were collected by some tools (such as Task Manager and 64-bit WinDbg) could not be opened in Visual Studio. This limitation has been removed in VS 2013.
- Visual Studio can debug dump files of native apps from ARM devices. Visual Studio can also debug apps dump files of managed apps from ARM devices, but only in the native debugger.
- To debug <u>kernel-mode</u> dump files in Visual Studio 2013, download the <u>Windows 8.1</u> <u>Version of Debugging Tools for Windows</u>. See <u>Kernel Debugging in Visual Studio</u>.
- Visual Studio can't debug dump files saved in the older dump format known as a <u>full</u> <u>user-mode dump</u>. Note that a full user-mode dump is not the same a dump with heap.
- To debug with the <u>SOS.dll (SOS Debugging Extension)</u> in Visual Studio, you must install the Debugging Tools for Windows that is part of the Windows Driver Kit (WDK). See <u>Windows 8.1 Preview: Download kits, bits, and tools</u>.
- Contents

Create a dump file

To create a dump file with Visual Studio:

- While you are debugging a process in Visual Studio, you can save a dump file when the debugger has stopped at an exception or at a breakpoint. Choose **Save Dump As**, **Debug**. In the **Save Dump As** dialog box, in the **Save as type** list, you can select **Minidump** or **Minidump with Heap** (the default).
- With <u>Just-In-Time Debugging in Visual Studio</u> enabled, you can attach the debugger to a crashed process that is running outside the debugger, and then save a dump file. See <u>Attach to Running Processes with the Visual Studio Debugger</u>

You can also create dump files with any program that supports the Windows minidump format. For example, the Procdump command-line utility from <u>Windows Sysinternals</u> can create process crash dump files based on triggers or on-demand. See <u>Requirements and</u> <u>limitations</u> in this topic for additional information about using other tools to create dump files.

• Contents

Open a dump file

- 1. In Visual Studio, choose File, Open, File.
- 2. In the **Open File** dialog box, locate and select the dump file. It will usually have a .dmp extension. Then choose **OK**.
- 3. The **Dump File Summary** window appears. In this window, you can view debugging summary information for the dump file, set the symbol path, start debugging, and copy the summary information to the clipboard.

AtiTest.dmp 🕂 🗙			
Minidump File 2/17/2011 11:36:17	Summary M		
 Dump Summary 			Actions
Dump FileAtlTestLast Write Time2/17/2Process NameAtlTestProcess Architecturex86Exception Code0x800Exception InformationA breatHeap InformationPresentError InformationVertex		mp : C:\Source\v 1 11:36:17 AM ee : d:\dd\sd\con 003 oint was encoun	 Debug with Native Only Set symbol paths Copy all to clipboard
 System Information OS Version 6.1.7600 CLR Version(s) ▲ Modules 			
Search		×	
Module Name	Module Version	Module Path	
AtlTest.exe ntdll.dll	0.0.0.0 6.1.7600.16385	d:\dd\sd\concord\ C:\Windows\SysW	

4. To start debugging, go to the Actions section, and choose either Debug with Native Only or Debug with Mixed.

Find binaries, symbol (.pdb) files, and source files

To use the full features of Visual Studio to debug a dump file, you need access to:

• The .exe file for which the dump was taken and other binaries (DLLs, etc.) that were used in the dump process.

If you are debugging a dump with heap data, Visual Studio can cope with missing binaries for some modules, but it must have binaries for enough modules to generate valid call stacks. Visual Studio includes the native modules in a dump file with heap.

- Symbol (.pdb) files for the .exe and other binaries.
- Source files for the modules that you are interested in.

The executable and the .pdb files must match exactly the version and build of the files used when the dump was created.

You can debug using the disassembly of the modules if you can't find the source files,

Default search paths for executable files

Visual Studio automatically searches these locations for executable files that aren't included in the dump file:

- 1. The directory that contains the dump file.
- 2. The path of the module that is specified in the dump file. This is the module path on the machine where the dump was collected.
- 3. The symbol paths specified in the **Debugging**, **Options**, **Symbols** page of the Visual Studio **Tools**, **Options** dialog box. You can add more locations to search on this page.

Using the No Binary / Symbol / Source pages

If Visual Studio can't find the files needed to debug a module in the dump, it displays an appropriate page (**No Binary Found**, **No Symbols Found**, or **No Source Found**). These pages provide detailed information about the cause of the issue and provide action links that can help you identify the correct location of the files. See <u>Specify Symbol (.pdb) and Source Files in the Visual Studio Debugger</u>.

Contents