Configuring Automatic Debugging

Article • 01/07/2021 • 4 contributors

In this article

Users can configure automatic debugging to help them determine why their system or an application has stopped responding.

Configuring Automatic Debugging for System Crashes

To configure the target computer to generate a crash dump file when the system stops responding, use the **System** application in Control Panel. Click **Advanced system settings**, which displays the **System Properties** dialog box. On the **Advanced** tab of that box, click **Settings** under **Startup and Recovery**, and then use the appropriate recovery options. Alternatively, you can configure crash dump options using the following registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\CrashControl

The file you can specify is the crash dump file. Its default name is Memory.dmp. You can debug a crash dump with a kernel-mode debugger, such as WinDbg or KD. For more information, see the documentation included with the debugger.

Configuring Automatic Debugging for Application Crashes

When an application stops responding (for example, after an access violation), the system automatically invokes a debugger that is specified in the registry for postmortem debugging, The process ID and event handle are passed to the debugger if the command line is properly configured. The following procedure describes how to specify a debugger in the registry.

To set a debugger as the postmortem debugger

1. Go to the following registry key:

   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug

2. Add or edit the **Debugger** value, using a REG_SZ string that specifies the command line for the debugger.

   The string should include the fully qualified path to the debugger executable. Indicate the process ID and event handle with "%ld" parameters to the debugger command line. Different debuggers may have their own parameter syntaxes for indicating these values.

Download PDF

When the debugger is invoked, the first "%ld" is replaced with the process ID and the second "%ld" is replaced with the event handle.

The following text is an example of how to setup up WinDbg as the debugger.

```
syntax                                                    Copy

"C:\debuggers\windbg.exe" -p %ld -e %ld -g
```

3. If you want the debugger to be invoked without user interaction, add or edit the **Auto** value, using a REG_SZ string that specifies whether the system should display a dialog box to the user before the debugger is invoked. The string "1" disables the dialog box; the string "0" enables the dialog box.

Excluding an Application from Automatic Debugging

The following procedure describes how to exclude an application from automatic debugging after the **Auto** value under the **AeDebug** key has been set to 1.

To exclude an application from automatic debugging

1. Go to the following registry key:

   HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\AeDebug

2. Add a REG_DWORD value to the **AutoExclusionList** subkey, where the name is the name of the executable file and the value is 1. By default, the Desktop Window Manager (Dwm.exe) is excluded from automatic debugging because otherwise a system deadlock can occur if Dwm.exe stops responding (the user cannot see the interface displayed by the debugger because Dwm.exe isn't responding, and Dwm.exe cannot terminate because it is held by the debugger).

   Windows Server 2003 and Windows XP: The **AutoExclusionList** subkey is not available; thus you cannot exclude any application, including Dwm.exe, from automatic debugging.

The default **AeDebug** registry entries can be represented as follows:

```
                                                          Copy

HKEY_LOCAL_MACHINE
   SOFTWARE
      Microsoft
         Windows NT
            CurrentVersion
               AeDebug
                  Auto = 1
                  AutoExclusionList
                     DWM.exe = 1
```

Related topics

Enabling Postmortem Debugging with WinDbg

Feedback

Additional resources

Training

Module

Interactively debug .NET apps with the Visual Studio debugger - Training

Learn how to efficiently debug your .NET app by using Visual Studio to fix your bugs quickly. Use the interactive debugger within Visual Studio to analyze and fix your C# applications.

Events

Nov 20, 12 AM - Nov 22, 12 AM

Gain the competitive edge you need with powerful AI and Cloud solutions by attending Microsoft Ignite online.

Register now