Welcome to Advanced .NET Debugging
Debugging Tools
Module Overview

Introduction to Debugging

- Problems in Production
- Challenges in debugging Production issues
- Production Environment Debugging Timeline
- Tools for .NET Debugging
- Review
Types of Problems in Production

System hang or deadlock
Fatal exception
Nonfatal exception
Data loss or inconsistency
Performance problems
Memory leaks or excessive memory usage
Causes of Problems in Production

Interaction with other systems
Hardware differences
Software differences
Unexpected load
Transient network conditions
Resource contention
Challenges of Debugging in Production

Real users
Pressure to resolve quickly
Reluctance to "make things worse"
Change control
Concern over tool installation
Challenges of Debugging in Production (Cont.)

Hard to access server
Can’t use normal tools
No repro
Infrequent problem
Hard to characterize problem
The problem only appears at a customer site
Production Debugging Timeline

Before a problem occurs

- Pre-install debugging tools in production
- Build symbols as appropriate
- Build in application exception handling and logging
- Design, build, and test for performance and scalability
- Build in monitoring and diagnostics
- Know the metrics of your system
- Find and resolve problems before they become crises
- Plan your deployments
Production Debugging Timeline (Cont.)

When a problem does occur
- Don’t panic; be methodical
- Evaluate the symptoms
- Document what you do
- Consider mitigation strategies
- Gather data

After the problem has occurred
- Analyze data
- Plan and implement steps to resolve
- Evaluate the improvement and repeat as necessary
Tools and environments

Environment
Production

Testing/Staging

Development

Least-invasive

Debugging tools
Debug Diagnostics, Perfmon, Event Logs, WER, Dr Watson, Netmon, procDump

Most Invasive

WinDbg, cdb, profilers, remote Debug with VS, SysInternals Suite,..
Production Debugging
Production Tools

**Event Logs**
- Detailed Information about application errors

**Performance Monitor**
- Diagnostic tool to collect System and Application performance data

**Dump Generation:**
- DrWatson, Windows Error Reporting (WER)
- ProcDump, Task Manager, Debug Diagnostics
**Dump Files**

**Kernel Mode Dump**

- Occurs when a kernel-mode error happens
- Different dump flavors available (Complete Memory, Kernel Memory, Small Memory, Automatic Memory)

---

```plaintext
*** Stop: 0x00000001E (0xF24A447A, 0x00000001, 0x00000000)
KMODE_EXCEPTION_NOT_HANDLED

*** Address F24A447A base at f24a0000, DateStamp 35825ef8d - wdmaud.sys

If this is the first time you’ve seen this Stop error screen, restart your computer. If this screen appears again, follow these steps:

Check to be sure you have adequate disk space. If a driver is identified in the Stop message, disable the driver or check with the manufacturer for driver updates. Try changing video adapters.

Check with your hardware vendor for any BIOS updates. Disable BIOS memory options such as caching or shadowing. If you need to use Safe Mode to remove or disable components, restart your computer, press F8 to select Advanced Startup Options, and then select Safe Mode.

Refer to your Getting Started manual for more information on troubleshooting Stop errors.

Kernel Debugger Using: COM2 (Port 0x2f8, Baud Rate 19200)
Beginning dump of physical memory
Physical memory dump complete. Contact your system administrator or technical support group.
```
USER.DMP File Flavors

**Mini-Dump:**
- Process and Module Information
- Call stack at the time of the crash for each thread of the user mode process
- Register settings
- Only for .NET 4.0 and above possible

**Mini-Dump with full memory**
- Mini Dump data
- full memory data, handle data, unloaded module information, basic memory information, and thread time information
- needed for .NET (1.0-3.5sp1)

**Full-Dump**
- All accessible committed pages of the application
Process bitness

Determine process bitness before debugging

32Bit OS
• 32Bit process

64Bit OS
• 32Bit or 64bit process

Depending on process bitness you choose the right debugging tools
Avoid creating a 64Bit Dump for a 32Bit Process

- You would get a dump of SysWow64

<table>
<thead>
<tr>
<th>Child-SP</th>
<th>RetAddr</th>
<th>Call Site</th>
</tr>
</thead>
<tbody>
<tr>
<td>00000000<code>000ce728 00000000</code>73a22bcd</td>
<td>wow64cpu!CpupSyscallStub+0x9</td>
<td></td>
</tr>
<tr>
<td>00000000<code>000ce730 00000000</code>73a9d07e</td>
<td>wow64cpu!Thunk0ArgReloadState+0x1a</td>
<td></td>
</tr>
<tr>
<td>00000000<code>000ce7f0 00000000</code>73a9c549</td>
<td>wow64!RunCpuSimulation+0xa</td>
<td></td>
</tr>
<tr>
<td>00000000<code>000ce840 00000000</code>76d684c8</td>
<td>wow64!Wow64LdrpInitialize+0x429</td>
<td></td>
</tr>
<tr>
<td>00000000<code>000ced90 00000000</code>76d67623</td>
<td>ntdll!LdrpInitializeProcess+0x17e2</td>
<td></td>
</tr>
</tbody>
</table>

- A lot of extensions will not work anymore

:000> !eeheap -gc
Failed to load data access DLL, 0x80004005
Verify that 1) you have a recent build of the debugger (6.2.14 or newer)

How do you know it is 32Bit?

- Use Task Manager (TaskMgr): *32 (Win 7) / Platform (Win 8/10)
ProcDump – Sysinternals

**Plus**

- Copy deployment install
- CPU spike monitoring
- Hang window monitoring (Detects missing message pumping)
- Unhandled exception monitoring
- Dumps based on performance counters.
- Clone support to minimize suspend time (>= Win7)
- Automatically creates a 32Bit Dump for a 32Bit Process on 64Bit OS

```
C:\Temp\SysinternalsSuite>procdump.exe
ProcDump v5.13 - Writes process dump files
Copyright (C) 2009-2013 Mark Russinovich
Sysinternals - www.sysinternals.com
With contributions from Andrew Richards

Monitors a process and writes a dump file when the process exceeds the
specified criteria or has an exception.

l counter threshold] [-r] [-t] [-d <callback DLL>] [<w] <process name or PID> [<
dump file> ! -i <dump file> ! -x <dump file> <image file> [arguments] >
```
**ProcDump – How to use**

Up to 3 dumps with full memory of process with ID 1234 when it exceeds 80% CPU usage for 5 seconds on one core

Directory: c:\temp

```
procdump -accepteula -ma -u -c 80 -s 5 -n 3
1234 c:\temp
```

Creates up to 2 Dumps with full memory on Process Termination and unhandled exception + Exceptions of type „System.Exception“ Directory: c:\temp within native mode.

```
procdump -accepteula -g -ma -n 2 -e 1 -f "System.Exception" -t 1234 c:\temp
```
Debug Diag

Data Collection
- Automatic Dump generation (based on triggers ex.: PerfCounter, Exceptions)
- IIS, COM+ Support
- Long time monitoring
- Import/Export of Rules
- Command Line Support

Data Analysis
- Analyze crash and memory issues
- Analyze Handle leaks
Components

DebugDiag is composed of the following components:

The Debugging Service - DwgSvc.exe
The Debugger Host - DwgHost.exe
The Collection UI - DebugDiag.Collection.exe
The Analysis UI - DebugDiag.Analysis.exe
The Rule Designer - DebugDiag.RuleDesigner.exe

DebugDiag uses 2 kinds of rules

Collection Rule
• .vbs script used by the debugging service and the debugger host

Analysis Rule
• .NET dll or XAML file used by the analysis engine.
• Replaces “Analysis Script” from version 1.x
Features

**Import/Export Collection Rules**

- .ddconfig Files – can be send with Mail
- Push to Live server(s) – can be imported once, then replicate to many

  `DebugDiag.exe /RemoveAllRules /ImportConfig myrules.ddconfig`

**Command Line**

- `xcopy + register.bat`
- `DebugDiag.exe /RemoveAllRules /ImportConfig myrules.ddconfig`
- `DbgHost.exe /attach MyRuleScript.vbs`
- `DbgHost.exe /dump /pn MyApp.exe`

**Performance Rules Analysis**

- Analyze a series of multiple dump files
- Find which functions take the most time/CPU

**Custom Rules Sets**

C:\Program Files\DebugDiag\Samples\AnalysisRules\DebugDiag.SampleAnalysisRules.sln
Collection

Collection Rules
• Crash Rule
• Performance Rule
• Memory and handle Leak Rule
• The auto-generated control scripts can be modified (samples are included)

Data
• Memory dumps (“Full” or “Mini”)
• Debugger logs (rule-generated stack traces, module/thread events, exception history)
• DbgSvc log (process start/exit history for entire machine, DebugDiag attach/detach history)
• Other - Event logs, IIS logs, .NET config files, etc.
• Data collection can be also triggered manually
Demonstration – Dump generation
Windows Error Reporting

Do you want to send more information about the problem?

Additional details about what went wrong can help Microsoft create a solution.

View Details  Send information  Cancel
Windows Error Reporting

Available since 2003 Server and Windows XP

- Background service
- Tracks and address errors relating to the OS
- For Windows components or applications

Plus

- It’s on the system, no installation
- Full Dumps up from 2008 Server or Vista SP1
- Can be used to collect data of your application and send it to Microsoft or your Server

Minus

- Limited options
- Config changes done via registry
Testing/Staging Tools

Profilers

- VS 2010..2015 Profiler + Windows Performance Toolkit
- PerfView, CLRProfiler, NP Profiler

Troubleshooting

- Sysinternal suite

Debuggers

- Windbg and cdb in conjunction with SOS
- Remote Debug using Visual Studio 2010..2015
- MDBG
PerfView

System wide ETW based Profiler
Memory (including Dumps), Disk and CPU Profiler
32Bit: Works great with 32Bit Applications
64Bit: JIT Frames break Callstack
  • use Windows 8 and above, Ngen, 32Bit Version
  • See
Comparing Debuggers

<table>
<thead>
<tr>
<th></th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visual Studio</td>
<td>GUI</td>
<td>Large memory footprint</td>
</tr>
<tr>
<td></td>
<td>Source level debugging</td>
<td>Not a free tool</td>
</tr>
<tr>
<td></td>
<td>Familiar</td>
<td>(Beyond Express or Community Version)</td>
</tr>
<tr>
<td></td>
<td>Build in .NET debugging</td>
<td>Requires installation</td>
</tr>
<tr>
<td></td>
<td>Edit &amp; Continue</td>
<td></td>
</tr>
<tr>
<td>WinDBG</td>
<td>GUI</td>
<td>Limited .NET debugging with</td>
</tr>
<tr>
<td></td>
<td>Source level debugging</td>
<td>SOS or psscorm extension</td>
</tr>
<tr>
<td></td>
<td>Designed specifically for debugging</td>
<td></td>
</tr>
<tr>
<td></td>
<td>XCopy deployment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Free tool</td>
<td></td>
</tr>
<tr>
<td>CDB</td>
<td>Command-Line</td>
<td>No source level debugging</td>
</tr>
<tr>
<td></td>
<td>Light weight</td>
<td>Limited .NET debugging with</td>
</tr>
<tr>
<td></td>
<td>Designed specifically for debugging</td>
<td>SOS or Psscorm extension</td>
</tr>
<tr>
<td></td>
<td>XCopy deployment</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Free tool</td>
<td></td>
</tr>
</tbody>
</table>
Visual Studio 2010/..../2015

- Possible to save and load dumps
- Managed Dump support through IL Interpreter (configurable) for .Net 4.0 or .NET 4.5 (>=VS2012)
  - Mini Dumps and Full Dumps
- Debugging Extension support within Immediate Window: .loadby sos clr
  - VS2012 or
  - VS2013 + Tools -> Options ->Debugging->Managed compatibility mode
To create a dump with Visual Studio 2010/..../2015

**VS2010:**
- Works using “native” debugging:
  - “Auto” in unmanaged C++
  - “Native Only”
  - “Mixed” in .NET

**>=VS2012:**
- Work per default
Visual Studio 2013

.NET Memory Analysis (Ultimate Edition)
Requires a dump, Supports CLR >= 4.5; Supports 64Bit Dumps for 32Bit Processes

Identify inefficient memory use
View size, and counts of all objects
View the largest individual instances of object types

Identify memory leaks
Ability to compare two dumps
See the reference graph preventing objects from being collected
Memory snapshots during debugging with update 4 (no command line interface)
New in Visual Studio 2015

- Compare memory consumption over time
  - Provides sizes and count for each type
  - Filters out objects marked for garbage collection (.NET)
- Works for both C++ and .NET apps
IntelliTrace

- What is IntelliTrace?
  Application recorder built into Visual Studio and Test Manager
  Can be used in development but also in production
  Included within Ultimate Edition

- "Historical" Debugging facility

- Options
  - Events only (approx. 20% of full trace)
  - Events and call information (can build custom)

- Limitations
  - Managed code only
  - Edit and continue is disabled
  - Needs Test Agent installation for “stand-alone“ in VS2010
IntelliTrace standalone

1. Install “IntelliTrace Collector for Visual Studio” to eg: C:\MyIntelliTraceCollector
   Expand the Cab File: expand /f:* IntelliTraceCollection.cab .
   Setup permissions if needed:

   icacls "C:\MyIntelliTraceCollector" /grant " <Domain\User> " :F

2. import „IntelliTrace PowerShell cmdlets” to collect data for Web or SharePoint applications:

   Import-Module
   "C:\IntelliTraceCollector\Microsoft.VisualStudio.IntelliTrace.PowerShell.dll"

   Trace your Application:

IntelliTraceSC.exe launch /cp:collection.xml /f:MyApp.itrace
C:\MyApp\MyApp.exe

*IIS Application Pool (triggers recycling)
Start-IntelliTraceCollection AppPoolName CollectionPlan LogFileDirectory
.. Stop-IntelliTraceCollection AppPoolName
Remote Debugging in VS

**Improvements**

- uses Windows Web Services API
- support from one domain to another, or to a work group
- symbols do not need to be copied to the remote computer (new for managed)
- „No Authentication“ mode for .NET and native
- Uses a single TCP/IP Port
- JavaScript Support

![Screenshot of Attach to Process dialog box with Windows Security alert]

- Enter your credentials
- Visual Studio was unable to create a secure connection to labs4016. Authentication failed.
- To retry, enter your credentials for labs.

**Options:**
- Use another account
- Connect a smart card

![Additional options for credentials]

- OK
- Cancel
.NET Source Code Debugging

Options

Task List
- Web Browser
- Projects and Solutions
- Source Control
- Text Editor
- Debugging
  - General
    - Edit and Continue
    - Just-In-Time
    - Output Window
    - Symbols
- IntelliTrace
- Performance Tools
- Database Tools
- F# Tools
- HTML Designer
- Office Tools
- Package Manager
- SOL Server Tools

General

- Enable breakpoint filters
- Enable the exception assistant
  - Unwind the call stack on unhandled exceptions
- Enable Just My Code
  - Show all members for non-user objects in variables windows (Visual Basic)
  - Warn if no user code on launch
- Enable .NET Framework source stepping
- Step over properties and operators (Managed only)
- Enable property evaluation and other implicit function calls
  - Call string-conversion function on objects in variables windows
- Enable source server support
  - Print source server diagnostic messages to the Output window
  - Allow source server for partial trust assemblies (Managed only)
- Highlight entire source line for breakpoints and current statement
- Require source files to exactly match the original version

OK | Cancel
Fuslogvw

Shows .NET Assembly binding issues
Comes with Windows SDK

*** Assembly Binder Log Entry (3/5/2007 @ 12:54:20 PM) ***
The operation failed. Bind result: hr = 0x80070002. The system cannot find the file specified...
--- A detailed error log follows.=== Pre-bind state information ===
LOG: DisplayName = graphicfailtest.resources, Version=0.0.0.0, Culture=en-US, PublicKeyToken=null
(Fully-specified)
LOG: Appbase = C:\Program Files\Microsoft.NET\FrameworkSDK\Samples\Tutorials\resourcesandlocalization\graphic\cs\LOG: Initial PrivatePath = NULL
LOG: Dynamic Base = NULL
LOG: Cache Base = NULL
LOG: AppName = NULL
Calling assembly : graphicfailtest, Version=0.0.0.0, Culture=neutral, PublicKeyToken...
SysInternals tool suite


Process Explorer
- “enhanced” Task Manager

Process Monitor
- Great Tool for monitoring File and Registry access
- Including Boot logging + Win32 Callstacks

Disk2VHD
- creates VHDs (Virtual PC) out of a running system
- >=Windows XP SP2, Windows Server 2003 SP1, and higher, including x64 systems.

PsExec
- Executes command within System context, event remotely, can be used to run Process Monitor after logoff

VMMMap
- Shows Memory segments in detail
Gflags - attaching the Debugger to A Process before the process starts

Add a value to the registry

- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Image File Execution Options
- Add the image name as a new subkey (notepad.exe)
- Add a new string value called Debugger
- Make string value ‘Windbg’ or ‘ntsd’

Can be done with gflags.exe as well
Gflags - attaching The Debugger to a “remote local” process

c:\debuggers\cdb.exe -server tcp:port=1234
BUILD ON

Demo

Gflags
Application Verifier

Runtime verification tool for unmanaged code assists in finding subtle programming errors that can be difficult to identify with normal application testing.

Default: Basics Checks

Shipped

- Microsoft.com/downloads
- Visual Studio 2005-2008 Team System for Developers
# Tools Resources

<table>
<thead>
<tr>
<th>Tool</th>
<th>use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gflags</td>
<td>debug application start, native memory corruptions</td>
</tr>
<tr>
<td>Adplus, procdump, Windows Error Reporting, Dr Watson, Debug Diag</td>
<td>Dump creation</td>
</tr>
<tr>
<td>Debug Diag 1.2 /2.0</td>
<td>Health Monitoring, Leaks, Crashes</td>
</tr>
<tr>
<td>Application Verifier</td>
<td>check native DLLs, Applications for leaks, Handle + Critical Section and other issues.</td>
</tr>
<tr>
<td>Fuslogvw</td>
<td>Managed DLLs load issues</td>
</tr>
<tr>
<td>Sxstrace Gflags</td>
<td>Native DLL load issues</td>
</tr>
<tr>
<td>PerfView</td>
<td>Memory and Performance Troubleshooting (Dump or Live)</td>
</tr>
</tbody>
</table>
## Tools Resources

<table>
<thead>
<tr>
<th>Tool</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>procdump</td>
<td><a href="http://www.sysinternals.com">http://www.sysinternals.com</a> – Tools Suite</td>
</tr>
<tr>
<td>Fuslogvw</td>
<td>Windows 7/8/10 SDK</td>
</tr>
<tr>
<td>Sxstrace</td>
<td>Shipped with Vista /2008 and above</td>
</tr>
</tbody>
</table>
Books:

- “Advanced .NET Debugging”
  By Mario Hewardt
  Addison-Wesley

- “Advanced Windows Debugging”
  by Mario Hewardt, Daniel Pravat
  Addison-Wesley

- „Debugging Applications“
  by John Robbins