CloudGen

ominidi verso l'evoluzione

TOPIC

Debugging a .NET program after crash

Post-mortem debugging

Microsoft® MVP Most Valuable Professional

@MircoVanini
www.proxsoft.it
https://www.linkedin.com/in/proxsoft

Thanks to our partners

CloudGen
ominidi verso l'evoluzione

AZUREBOX

ADAWEN
Add Value Group

BETA 80 GROUP

elettrone
intelligenza energetica

jmatica
BUSINESS INNOVATION TECHNOLOGIES

fairmat

JET BRAINS

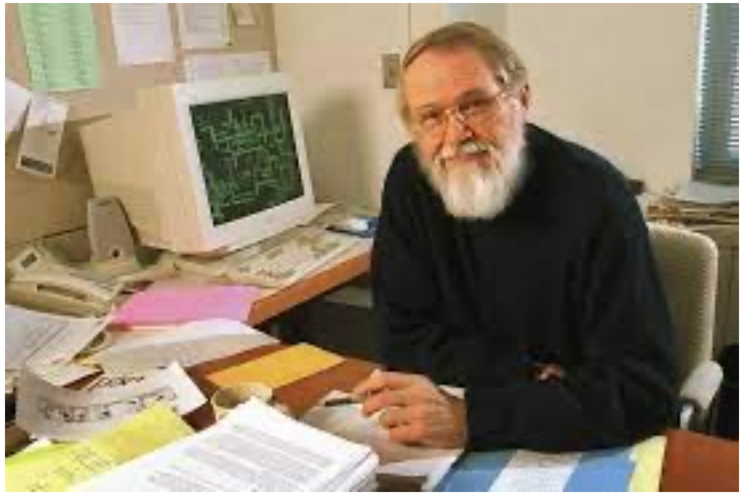
Microsoft

Packt>

Software Bugs are Expensive

“Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.”

Brian Kernighan



Importance of debugging

- Perfect code is an illusion
- Legacy Code
- Deeper Understanding
- Helps you learn & write better code in the future

The three debugging phases

- Isolation
- Replication
- Fix

Production Debugging

Requirements

Obtain actionable information about crashes and errors

Obtain accurate performance information

Limitations

Can’t install Visual Studio

Can’t suspend production servers

Can’t run intrusive tools

Dump File

- A user dump is a snapshot of a running process
- A kernel dump is a snapshot of the entire system
- Dump files are useful for post-mortem diagnostics and for production debugging
- Anytime you can’t attach and start live debugging, a dump might help

Limitations of Dump Files

- **A dump file is a static snapshot**
  - You can’t debug a dump, just analyze it
  - Sometimes a repro is required (or more than one repro)
- Sometimes several dumps must be compared

Taxonomy of Dumps

- Crash dumps are dumps generated when an application crashes
- Hang dumps are dumps generated on-demand at a specific moment
- These are just names, the contents of the dump files are the same!

Generating a Hang Dump

Task Manager

right-click and choose “Create Dump File”
Creates a dump in **%LOCALAPPDATA%\Temp**

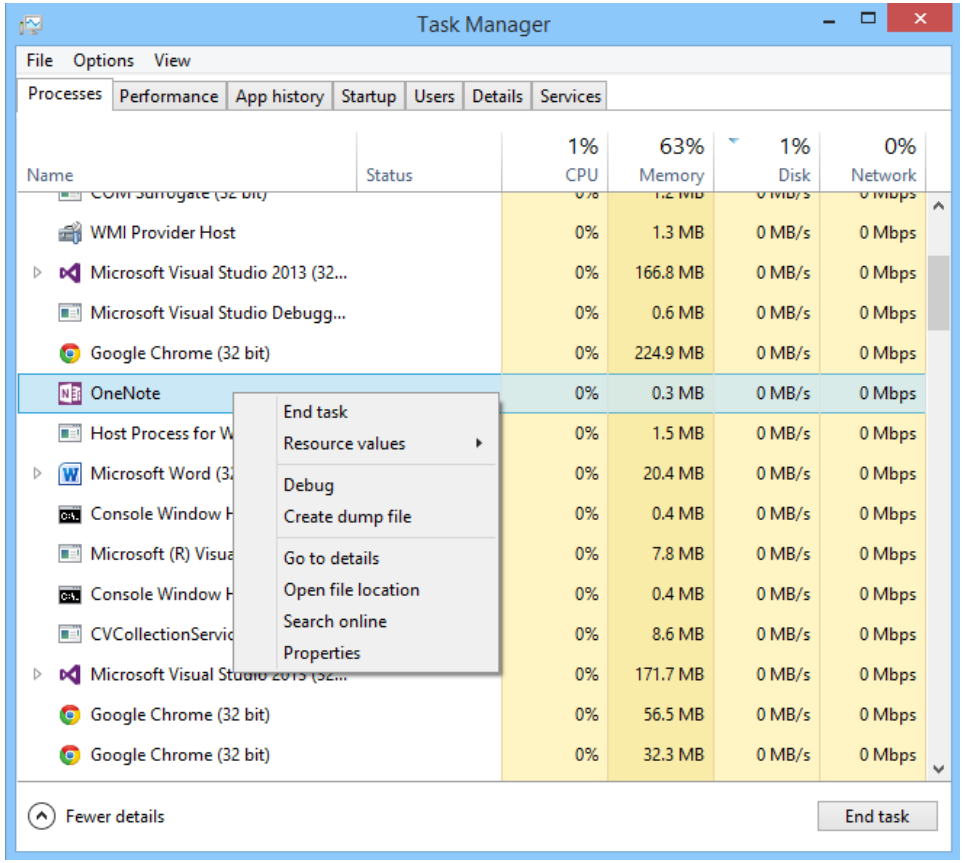
SysInternals - Procdump

Sysinternals utility for creating dumps,

Light-weight, no-install utility for generating dumps

DebugDiag

Microsoft tool for monitoring and dump generation

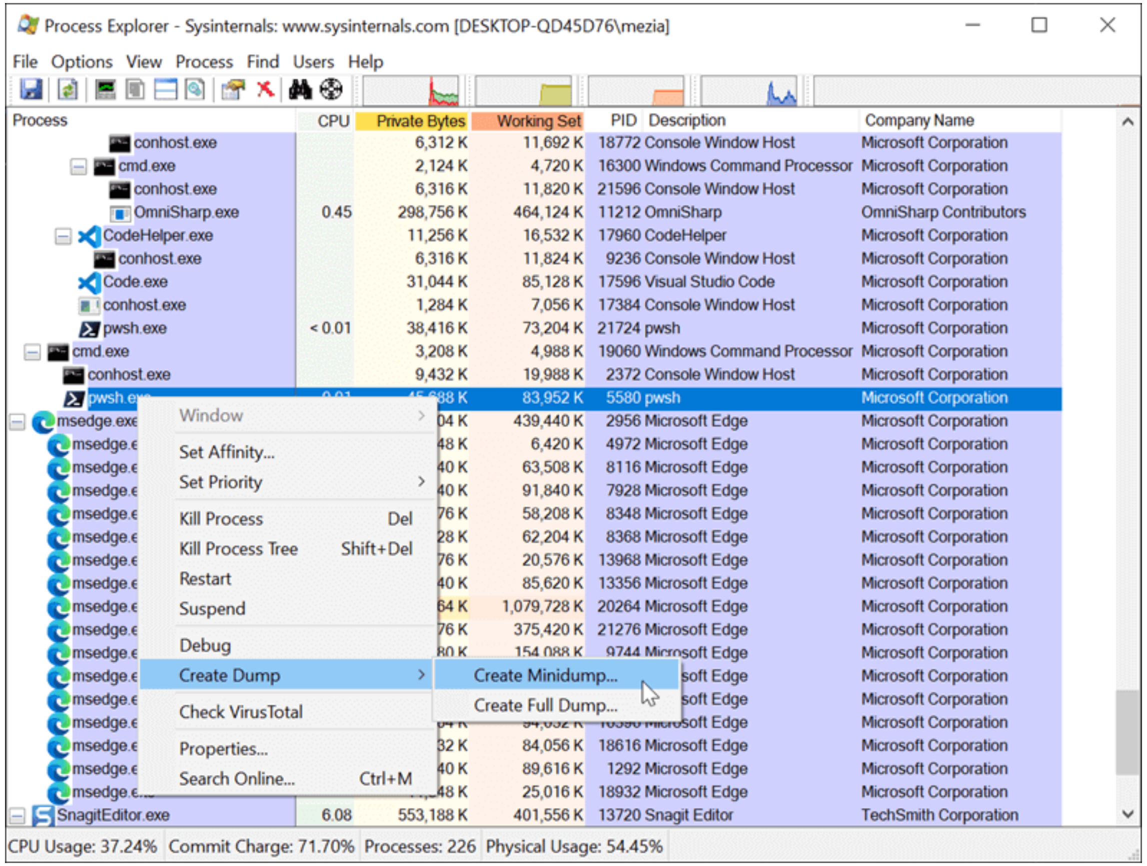


Name	Status	1% CPU	63% Memory	1% Disk	0% Network
COM Surrogate (32 bit)		0%	1.2 MB	0 MB/s	0 Mbps
WMI Provider Host		0%	1.3 MB	0 MB/s	0 Mbps
Microsoft Visual Studio 2013 (32...		0%	166.8 MB	0 MB/s	0 Mbps
Microsoft Visual Studio Debugg...		0%	0.6 MB	0 MB/s	0 Mbps
Google Chrome (32 bit)		0%	224.9 MB	0 MB/s	0 Mbps
OneNote		0%	0.3 MB	0 MB/s	0 Mbps
Host Process for W		0%	1.5 MB	0 MB/s	0 Mbps
Microsoft Word (32		0%	20.4 MB	0 MB/s	0 Mbps
Console Window H		0%	0.4 MB	0 MB/s	0 Mbps
Microsoft (R) Visua		0%	7.8 MB	0 MB/s	0 Mbps
Console Window H		0%	0.4 MB	0 MB/s	0 Mbps
CVCollectionServic		0%	8.6 MB	0 MB/s	0 Mbps
Microsoft Visual Studio 2013 (32...		0%	171.7 MB	0 MB/s	0 Mbps
Google Chrome (32 bit)		0%	56.5 MB	0 MB/s	0 Mbps
Google Chrome (32 bit)		0%	32.3 MB	0 MB/s	0 Mbps

Generating a Hang Dump

SysInternals - Process Explorer

Right-click on the process and select the "Create Dump" menu item



Process Explorer - Sysinternals: www.sysinternals.com [DESKTOP-QD45D76\mezia]

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
conhost.exe		6,312 K	11,692 K	18772	Console Window Host	Microsoft Corporation
cmd.exe		2,124 K	4,720 K	16300	Windows Command Processor	Microsoft Corporation
conhost.exe		6,316 K	11,820 K	21596	Console Window Host	Microsoft Corporation
OmniSharp.exe	0.45	298,756 K	464,124 K	11212	OmniSharp	OmniSharp Contributors
CodeHelper.exe		11,256 K	16,532 K	17960	CodeHelper	Microsoft Corporation
conhost.exe		6,316 K	11,824 K	9236	Console Window Host	Microsoft Corporation
Code.exe		31,044 K	85,128 K	17596	Visual Studio Code	Microsoft Corporation
conhost.exe		1,284 K	7,056 K	17384	Console Window Host	Microsoft Corporation
pwsh.exe	< 0.01	38,416 K	73,204 K	21724	pwsh	Microsoft Corporation
cmd.exe		3,208 K	4,988 K	19060	Windows Command Processor	Microsoft Corporation
conhost.exe		9,432 K	19,988 K	2372	Console Window Host	Microsoft Corporation
pwsh.exe	0.01	45,688 K	83,952 K	5580	pwsh	Microsoft Corporation
msedge.exe		04 K	439,440 K	2956	Microsoft Edge	Microsoft Corporation
msedge.e		48 K	6,420 K	4972	Microsoft Edge	Microsoft Corporation
msedge.e		40 K	63,508 K	8116	Microsoft Edge	Microsoft Corporation
msedge.e		40 K	91,840 K	7928	Microsoft Edge	Microsoft Corporation
msedge.e		76 K	58,208 K	8348	Microsoft Edge	Microsoft Corporation
msedge.e		28 K	62,204 K	8368	Microsoft Edge	Microsoft Corporation
msedge.e		76 K	20,576 K	13968	Microsoft Edge	Microsoft Corporation
msedge.e		40 K	85,620 K	13356	Microsoft Edge	Microsoft Corporation
msedge.e		64 K	1,079,728 K	20264	Microsoft Edge	Microsoft Corporation
msedge.e		76 K	375,420 K	21276	Microsoft Edge	Microsoft Corporation
msedge.e		80 K	154,088 K	9744	Microsoft Edge	Microsoft Corporation
msedge.e		[illegible]	[illegible]	[illegible]	soft Edge	Microsoft Corporation
msedge.e		[illegible]	[illegible]	[illegible]	soft Edge	Microsoft Corporation
msedge.e		[illegible]	[illegible]	[illegible]	soft Edge	Microsoft Corporation
msedge.e		32 K	84,056 K	18616	Microsoft Edge	Microsoft Corporation
msedge.e		40 K	89,616 K	1292	Microsoft Edge	Microsoft Corporation
msedge.e		48 K	25,016 K	18932	Microsoft Edge	Microsoft Corporation
SnagitEditor.exe	6.08	553,188 K	401,556 K	13720	Snagit Editor	TechSmith Corporation

CPU Usage: 37.24% Commit Charge: 71.70% Processes: 226 Physical Usage: 54.45%

Generating a Hang Dump

.NET Core diagnostic global tools

dotnet-dump

The dotnet-dump tool is a way to collect and analyze Windows and Linux core dumps without a native debugger.

dotnet-gcdump

The dotnet-gcdump tool is a way to collect GC (Garbage Collector) dumps of live .NET processes.

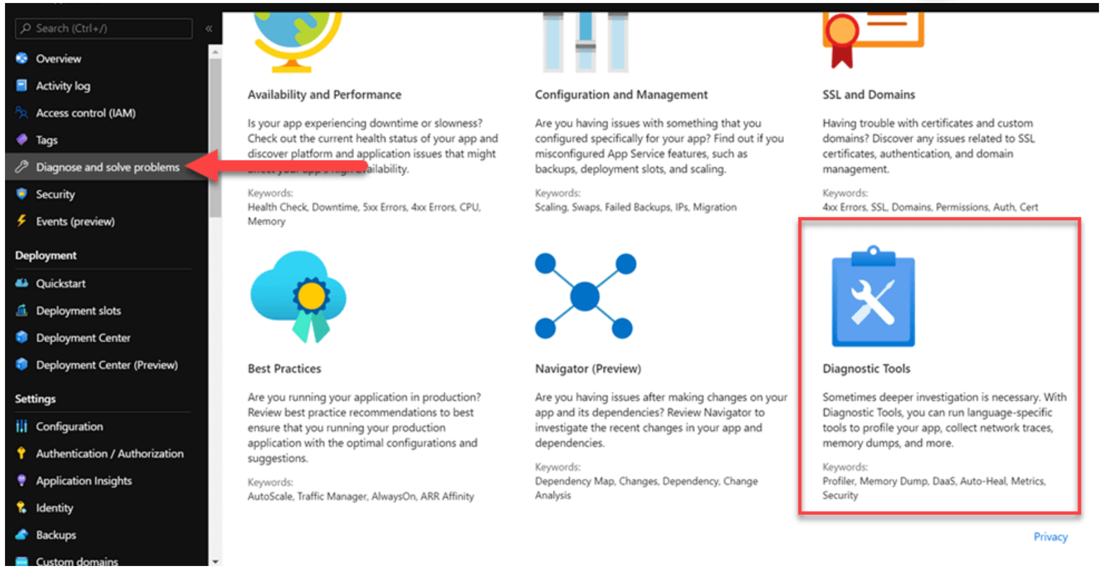
Generating a Hang Dump

Azure App Services

Select your App Service
Go to "Diagnose and solve problems"
Select Diagnose Tools

Select "Collect Memory Dump"
Click on the "Collect Memory Dump" button
After a few minutes, the dump
is available in the configured storage account

[Collect and Automate Diagnostic Actions with Azure App Services](#)



Availability and Performance

Is your app experiencing downtime or slowness? Check out the current health status of your app and discover platform and application issues that might [illegible]ailability.

Keywords:
Health Check, Downtime, 5xx Errors, 4xx Errors, CPU, Memory

Configuration and Management

Are you having issues with something that you configured specifically for your app? Find out if you misconfigured App Service features, such as backups, deployment slots, and scaling.

Keywords:
Scaling, Swaps, Failed Backups, IPs, Migration

SSL and Domains

Having trouble with certificates and custom domains? Discover any issues related to SSL certificates, authentication, and domain management.

Keywords:
4xx Errors, SSL, Domains, Permissions, Auth, Cert

Best Practices

Are you running your application in production? Review best practice recommendations to best ensure that you running your production application with the optimal configurations and suggestions.

Keywords:
AutoScale, Traffic Manager, AlwaysOn, ARR Affinity

Navigator (Preview)

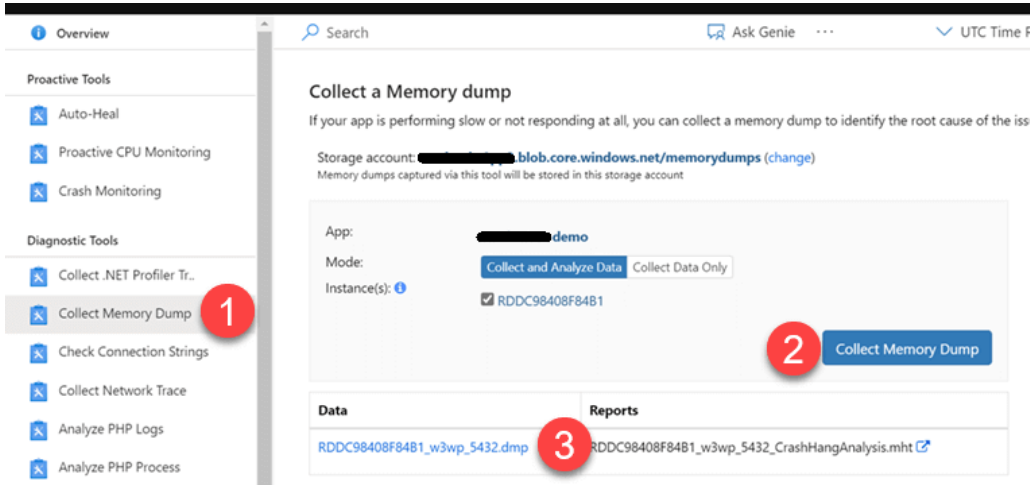
Are you having issues after making changes on your app and its dependencies? Review Navigator to investigate the recent changes in your app and dependencies.

Keywords:
Dependency Map, Changes, Dependency, Change Analysis

Diagnostic Tools

Sometimes deeper investigation is necessary. With Diagnostic Tools, you can run language-specific tools to profile your app, collect network traces, memory dumps, and more.

Keywords:
Profiler, Memory Dump, DaaS, Auto-Heal, Metrics, Security



Collect a Memory dump

If your app is performing slow or not responding at all, you can collect a memory dump to identify the root cause of the iss

Storage account: [redacted].blob.core.windows.net/memorydumps (change)
Memory dumps captured via this tool will be stored in this storage account

App: [redacted]demo
Mode: [✓ Collect and Analyze Data] Collect Data Only
Instance(s):
☑ RDDC98408F84B1

Collect Memory Dump

Data	Reports
RDDC98408F84B1_w3wp_5432.dmp	RDDC98408F84B1_w3wp_5432_CrashHangAnalysis.mht

Analyzing Dumps File

Native debugger (WinDBG)

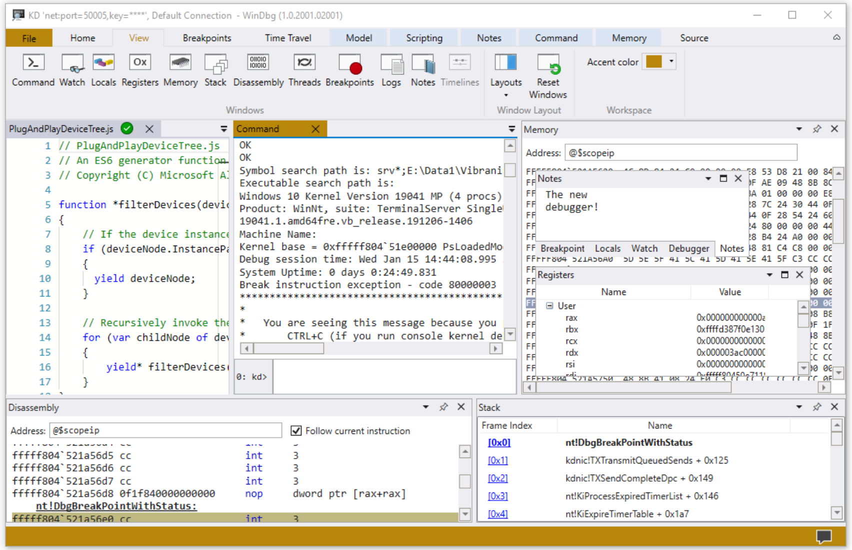
Analyze crash dump files by using WinDbg

DebugDiag

How to Use Debug Diagnostics to Analyze a Memory Dump

Visual Studio

Dump files in the Visual Studio debugger



```
// PlugAndPlayDeviceTree.js
// An ES6 generator function
// Copyright (C) Microsoft A

function *filterDevices(devic
{
    // If the device instance
    if (deviceNode.InstancePa
    {
      yield deviceNode;
    }

    // Recursively invoke the
    for (var childNode of dev
    {
       yield* filterDevices(
    }
```

```
OK
OK
Symbol search path is: srv*;E:\Data1\Vibrani
Executable search path is:
Windows 10 Kernel Version 19041 MP (4 procs)
Product: WinNt, suite: TerminalServer Single
19041.1.amd64fre.vb_release.191206-1406
Machine Name:
Kernel base = 0xfffff804`51e00000 PsLoadedMo
Debug session time: Wed Jan 15 14:44:08.995
System Uptime: 0 days 0:24:49.831
Break instruction exception - code 80000003
*********************************************
*
*   You are seeing this message because you
*       CTRL+C (if you run console kernel de
0: kd>
```

Address: @$scopeip

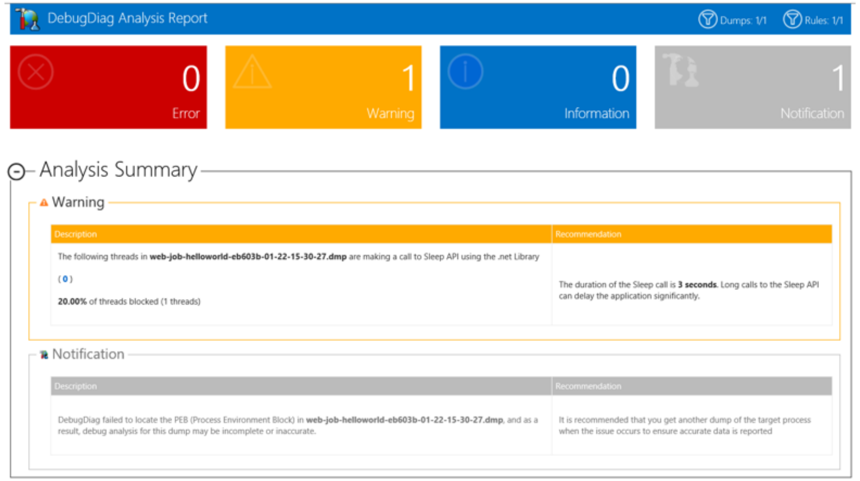
Notes: The new debugger!

Name	Value
⊟ User	
rax	0x000000000000a
rbx	0xffffd387f0e130
rcx	0x000000000000C
rdx	0x000003ac0000C
rsi	0x000000000000C

Address: @$scopeip ☑ Follow current instruction

```
fffff804`521a56d5 cc              int     3
fffff804`521a56d6 cc              int     3
fffff804`521a56d7 cc              int     3
fffff804`521a56d8 0f1f840000000000 nop    dword ptr [rax+rax]
  nt!DbgBreakPointWithStatus:
fffff804`521a56e0 cc              int     3
```

Frame Index	Name
[0x0]	**nt!DbgBreakPointWithStatus**
[0x1]	kdnic!TXTransmitQueuedSends + 0x125
[0x2]	kdnic!TXSendCompleteDpc + 0x149
[0x3]	nt!KiProcessExpiredTimerList + 0x146
[0x4]	nt!KiExpireTimerTable + 0x1a7



0 Error

1 Warning

0 Information

1 Notification

Analysis Summary

Warning

Description	Recommendation
The following threads in **web-job-helloworld-eb603b-01-22-15-30-27.dmp** are making a call to Sleep API using the .net Library (0) **20.00%** of threads blocked (1 threads)	The duration of the Sleep call is **3 seconds**. Long calls to the Sleep API can delay the application significantly.

Notification

Description	Recommendation
DebugDiag failed to locate the PEB (Process Environment Block) in **web-job-helloworld-eb603b-01-22-15-30-27.dmp**, and as a result, debug analysis for this dump may be incomplete or inaccurate.	It is recommended that you get another dump of the target process when the issue occurs to ensure accurate data is reported

Null reference exceptions

GC Heap pressure, OOM Exceptions

Stack overflow

Dead Lock

Threadpool OutOfThreads

Common Bugs

CRASHES

- Check the event viewer
- Capture dump on crash
- Look at the faulting stack

PERFORMMANCE ISSUES

- Capture one or more dumps
- Look at all stacks
- if you can repro in test, consider profiling

- Low CPU
  Waiting for an external resource
  Deadlock

- High CPU
  Tight loop, High CPU in GC

MEMORY LEAKS

- Capture multiple dumps
- Compare to see what objects are leaking
- Find out why they are still around

About me

Mirco Vanini

Microsoft MVP Developer Technologies

Consultant focused on industrial and embedded solutions using .NET and other native SDKs with over 30 years of experience, XeDotNet community co-founder, speaker and Microsoft MVP since 2012

MVP Microsoft® Most Valuable Professional

@MircoVanini
www.proxsoft.it
https://www.linkedin.com/in/proxsoft

#CodeGen2024

@cloudgen_verona

Thank you

Any questions?

@MircoVanini
www.proxsoft.it
https://www.linkedin.com/in/proxsoft