

# **Debugging on the Windows Platform**

**(including .NET Framework 4.0)**

**Daniel Bullington**

**Sr. Application Developer/Lead, VHDA**

**<http://blog.softwareishardwork.com>**

# Orders of the Day

- Debugging Basics, APIs, and Support
- Making Debugging Easier
- Visual Studio Debugging Tips
- .NET 4.0 Debugging Changes

# Debugging Basics, APIs, and Support

# Debugging Landscape

- The process of analyzing software or hardware for the root cause of:
  - Crashes, hangs, faulty results, security vulnerabilities, or poor performance or lack of scalability.
- The debugging experience varies greatly from platform to platform, runtime to runtime.

# Debug Support on Windows

- Debugging support depends on the runtime:
  - Native
    - C/C++ including COM, ATL, MFC
  - .NET Framework (a.k.a Managed)
    - CLR targeted languages such as C#, VB.NET, etc.
  - Other (*not discussed*)
    - Database (T-SQL, PL-SQL, etc.)
    - Scripting (PS, VBS, JS, etc.)
    - Others Runtimes (JRE, VB6, etc.)

# Who Does What?

- The *Debuggee* is the process being debugged.
- The *Debugger* is the debugging process.
- A *debugging loop* describes the logic in which a debugger and debuggee interact...

# Native Debug Support

- Native debugging support is a kernel mode phenomenon.
- Support is realized as a simple API requiring no support from the debuggee.
- Out of process debugging via a per process “pipeline”.

# .NET v2.0 Debug Support

- .NET debugging support is a user mode phenomenon.
- Support is realized in-process (with respect to the debuggee) within the CLR using IPC events.
- Can debug running processes only!
- The API is rich and complex
  - ICorDebug, et al.

# Debug APIs

- Useful if you are writing your own debugger or extending existing debuggers.
  - Google for sample code
- Since excellent debuggers are already offered by MS, lets not reinvent the wheel...

# Native Debuggers

- Share the same debug engine and are production quality:
- KD
  - Command line kernel mode debugger
- CDB/ NTSD
  - Command line user mode debugger
  - Both are nearly identical.
  - NTSD spawns a new console; CDB inherits console.
- WinDBG
  - GUI front-end to all of the above.
- These debuggers are very powerful but complex.
- Debug live running processes or process memory dumps.
- See DTfW documentation.

# .NET Debuggers

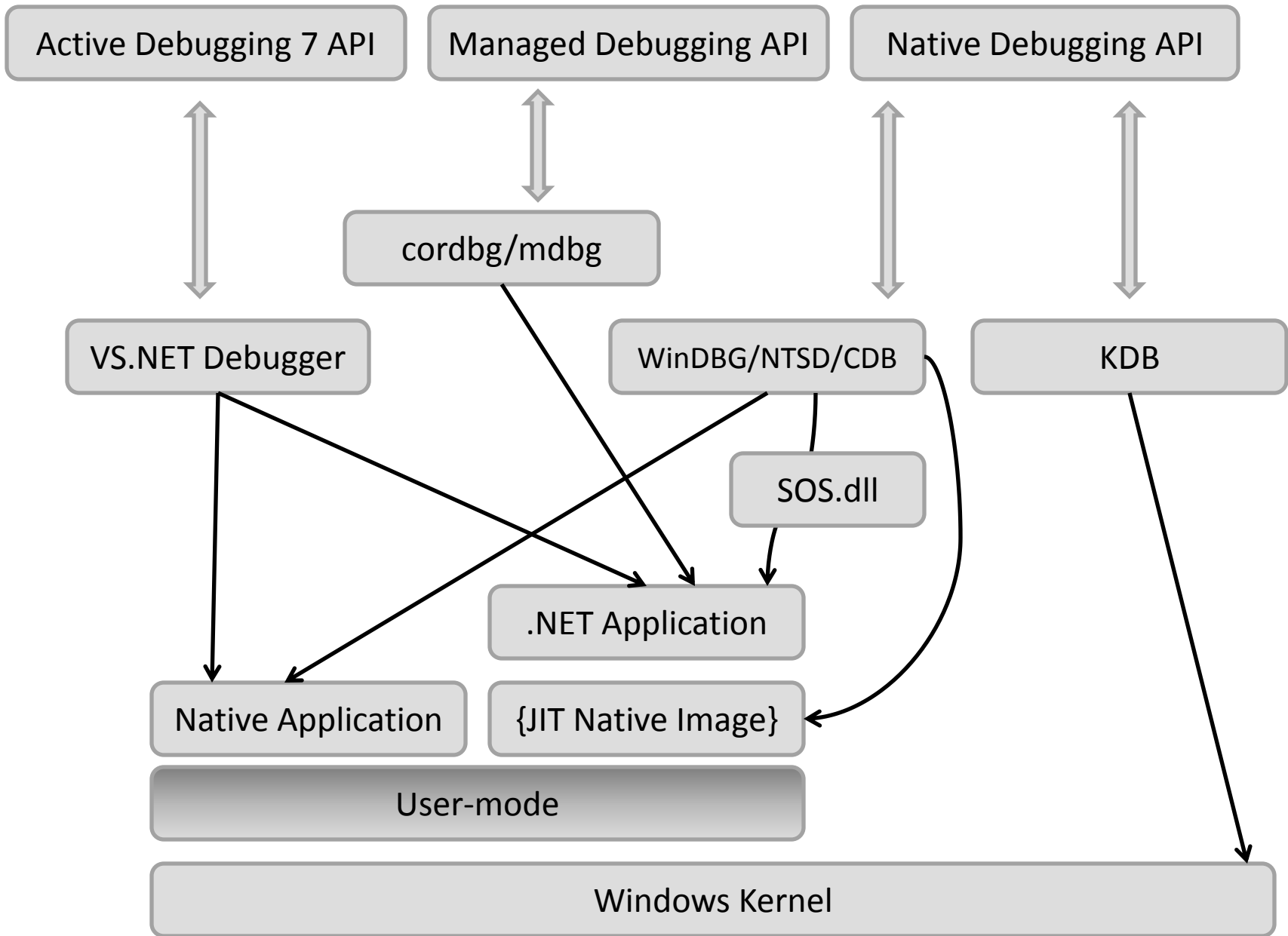
- `cordbg`
  - Runtime command line debugger written in unmanaged C++ leveraging ICorDebug (COM); this is considered obsolete.
- `mdbg`
  - Runtime command line debugger written in C# and used COM interop to leverage ICorDebug. Not intended as a production debugger.
- `dbgclr`
  - Visual Studio.NET Debugger “lite”; no longer shipped as of VS 2008.
- WinDBG
  - More on this in just a second...

# Visual Studio.NET Debugger

- Do not install on a production server.
- Remote debug production systems (sketchy).
- Debugs native, managed, script, and T-SQL code.
- Exposes the Active Debugging 7 API.

# WinDBG + SOS => CLR ?

- Very useful as production debuggers for .NET code.
- WinDBG has no knowledge of .NET.
- We can leverage the SOS.dll WinDBG extension (*sos.dll*) to bridge the gap.
  - A "prototype" managed debugging API
- Cannot single step through .NET code
- You CAN analyze a live .NET process or memory dump.



# Making Debugging Easier

# Debug Symbols

- Regardless of the runtime (native, managed), always set your compiler to generate debug symbols.
- PDBs are the key to single stepping code.
- PDBs are the key to native stack traces.
- PDBs are NOT key to managed stack traces.

# Build Flavor

- This is the one of the most misunderstood settings in the MS development universe.
- Debug disables certain optimizations to allow for easier debugging at the expense of performance.
- Release mode enables certain optimizations (such as method in lining) at the expense of debugging.
  - Single stepping in release mode
    - Optimizations can cause odd results when constructs are “optimized away”

# Build Platform

- Make sure to select the proper build platform for the code and runtime you are targeting.
- For managed code, choose AnyCPU unless you are using P/Invoke.
- For unmanaged code, multiple builds are required to targets different architectures.

# Binary Versioning

- Always version your binaries.
  - Unmanaged: embedded resources.
  - Managed: assembly attributes.

# Binary Signing

- Ensures integrity and identity of binaries:
  - Digital signatures
  - Assembly strong naming (managed only)

# Visual Studio Debugging Tips

# Better Debugger Options

- *Tools > Options > Debugger*
- Disable the “execution assistant” feature.
  - Just “buggy” as hell.
- Disable “just my code” feature.
  - Prevents hiding of non-solution code from stack window.
- Disable “step over properties and operators” feature.
  - Prevents hiding property code from you.
- Disable “call string conversion function on objects in variables window” feature.
  - Prevents side effects of debugging.

# Make Object ID

- Right click a non-null watch variable and choose “Make Object ID” (C# only) in debug stop.
- Track an object lifetime even when watch variable goes out of scope.
- Very powerful feature.

# v2.0 Mixed Mode Debugging

- On a 32-bit x86 machine, you can debug both the native and managed sides of an application using VS.NET debugger.
- On a 64-bit x64 machine, you have to spin up two separate debuggers for each side.

# Exception Breakpoints

- *Debug > Execptions...*
- Allows you to break on a certain exceptions when they occur, even if an exception handler could trigger a swallow.
  - Called a first chance exception.
- A second chance exception is an unhandled exception.
- Managed and native (i.e. SEH, Cxx, COM) support.

# .NET v4.0 Debugging Changes

# .NET v4.0 Debug Support

- .NET debugging support now leverages the native debugging “pipeline”.
- Support is realized out-of-process (with respect to the debuggee) within the CLR using native debugging “pipeline”.
- The API is rich and complex
  - Self-consistent debugging API based on ICorDebug
  - Operates in two difference modes - "pure out-of-process v4" and "v2 compatibility”.

# .NET v4.0 Debug Support (cont'd)

- Today, a pure v4 usage only has the ability to do inspection operations.
  - Dump debugging in VS2010 is big win!
- But more out-of-process functionality (like stepping) in future releases.
- Debugging a v4.0 managed app requires a v4.0-aware debugger.
- Support for 64-bit mixed mode debugging!

# Bonus Track: A Profiling API Change

- Ability to attach profiler to running process.
- Inspect what locks a thread holds and what locks a thread is blocked on in the debugger, which is brought to you by the previous bullet point.

# Demo: VS2010 Debugging

# References

- MSDN
- Tess Fernandez (Microsoft)
- Mike Stall (Microsoft)
- Mark Russinovich (Microsoft)
- John Robbins (Wintellect)
- Rick Byers (Microsoft)