Orasi – Debugging

Intermountain Healthcare

Course Goals

Goals

- • Understand what a crash is
- • Use tools to analyze crash
- • Gather information to file defect

Non-Goals

- • Debugging expertise

Overview

- To discover the root cause of an unhandled exception we can use the Microsoft systems debuggers, cdb or windbg, to debug the process.
- We should have symbols for each component we are debugging.
- Cdb can be used to:
  - Get the call stack of an unhandled exception (crash) that is occurring within the process.
  - Find other issues that are not detected without cdb attached such as first chance access violations (AV) that are hidden bugs.
  - Provide memory dumps (.dmp files) for developers to debug offline.
  - Find and debug memory leaks once you know they already exist. This is a more advanced topic but nonetheless, useful.

CrashApp

- File, New, Project
- Visual C#, Windows, Windows Forms Application
- Location: c:\Orasi\VS\Projects
- Name: CrashApp (or whatever)

Crash App Controls

- Toolbox (floating on left)
  - Common controls, Button
    - Properties (F4)
    - Text: Null Ref
    - (Name): buttonNullRef
  - Double-Click on buttonNullRef to get to code.

Crash App – Null Ref Code

```
FileStream f = null;
string path = @"c:
\orasi\VS\Projects\Data\Customer.txt";
if (File.Exists(path))
{
    f = File.OpenRead(path);
}
f.Close();
```

CrashApp - Run

- Run from VS – F5
- Make it fail
- Run from Windows File Explorer

CrashApp – Stack Overflow code

```
private void button2_Click(…)
{
    f1();
}

private void f1()
{
    f1();
}
```

Setting up Debuggers

- Install cdb on the machine that will be running the process that will be monitored:
- http://msdn.microsoft.com/en-US/windows/hardware/hh852363
  - Click Install and Download
  - Ignore that it mentions Windows 8
  - Uncheck every option except
    - Windows Performance Toolkit
    - Debugging Tools for Windows

Configuring System PATH

- We need to add the debugging directory to the System Path which sets the symbols path permanently
- In Start menu, right click on Computer, choose Properties, Advanced System Settings, Environment Variables…, System variables
- Find variable name Path.
- Click Edit, then hit the **End** key to go to the end of the line.
- Carefully append the following text that begins with a semi-colon:
  - ;C:\Program Files (x86)\Windows Kits\8.0\Debuggers\x86

_nt_symbol_path - Temporary

- Set the default symbols path for operating system symbols
- There is an environment variable named **_nt_symbol_path** that all Microsoft tools look for. Yes, the path begins with an underscore, and if you don't include it, it won't work.
- Set the Symbol path from the command line
- When debugging from the command line the following command can be used with semi-colon delimiters between paths.
  - set _nt_symbol_path=srv*http://msdl.microsoft.
    com/download/symbols

_nt_symbol_path - Permanent

- Set the symbols path permanently
- Right click on My Computer, Properties, Advanced System Settings, Environment Variables, System, New
- Variable name: _nt_symbol_path
- Variable value:
  - srv*http://msdl.microsoft.com/download/symbols

Using Cdb - Connecting

- Cdb /?
- Cdb –pn crashapp.exe
- Cdb -g –G –pn crashapp.exe
- Cdb –p 992
- Cdb –g –G crashapp.exe

Using Cdb - Disconnecting

- Ctrl + C – to break in
- To quit and terminate the application
  - q - quit
- To detach and leave the application running
  - .detach – to disconnect
  - q – to quit

Using Cdb – create dump file

- Ctrl + C
- .dump /ma c:\orasi\Crashapp1.dmp
- Exit
  - To quit and terminate the application
    - q - quit
  - To detach and leave the application running
    - .detach – to disconnect
    - q – to quit

Using Cdb – debug dump file

- Cdb –z c:\orasi\Crashapp1.dmp
- .dumpcab –a c:\orasi\Crashapp1.cab
- q – to quit

Using cdb – help commands

- When debugging a process or dump file:
  - ? – help
  - .help – dot commands help
  - !help – extensions help
  - .hh – load windows help file

Using Cdb - commands

- lm – load modules (list modules)
- Lm v mclr – show info about module (clr)
- .sympath – show symbols path
- | - show process information (pipe)
- .reload – reload symbols
- .reload /f – force reload symbols
- .reload /f clr – force reload one module

Using Cdb - Logging

- .logopen c:\orasi\Crashapp1.log
- .logclose
- .logappend c:\orasi\Crashapp1.log

Using Cdb - commands

- Kb – stack backtrace
- ~ - list all native threads
- ~*kb – list stacks on all threads
- ~3s – change to thread id 3

Using Cdb - Managed

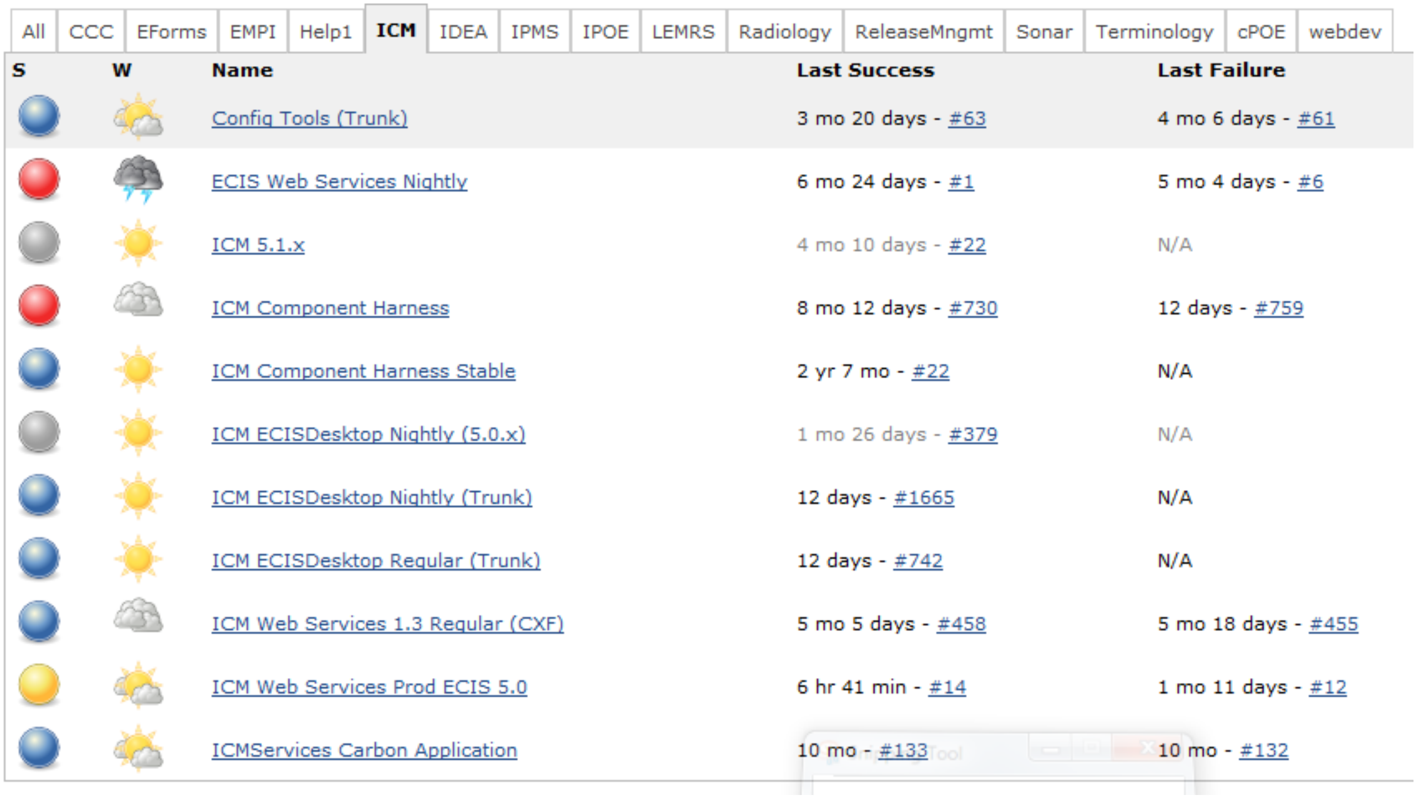
- .loadby sos clr – Loads add-in SOS (Son of Strike)
- !sos.help – SOS help
- !threads
- !dumpstack
- !clrstack
- !analyze –v

Setting up Symbols

- You can end up spend more time getting correct symbols than debugging the failure.
- Symbols are needed at debugging time to determine the root cause of a failure or memory leak.
- **Symbols are not needed at runtime**, meaning that while the test is running, the symbols are **not needed**.
- They are only needed after a failure has occurred and you intend to debug the failure.
- I usually don’t debug a failure on the machine that the failure occurred on; we can create a dump file of the failure and debug the dump file at a later time on another system.
- This allows your test machines to keep finding new failures and saving dump files, while you debug them at your leisure.
- I explain the dump commands below, so pay special attention.

Getting Intermountain Symbols

- Symbols are provided on “Jenkins”. Go to the Jenkins Url below.
- http://lpv-ecismig01:8080/view/ICM%20/



All | CCC | EForms | EMPI | Help1 | **ICM** | IDEA | IPMS | IPOE | LEMRS | Radiology | ReleaseMngmt | Sonar | Terminology | cPOE | webdev

S	W	Name	Last Success	Last Failure
🔵	⛅	Config Tools (Trunk)	3 mo 20 days - #63	4 mo 6 days - #61
🔴	⛈	ECIS Web Services Nightly	6 mo 24 days - #1	5 mo 4 days - #6
⚪	☀	ICM 5.1.x	4 mo 10 days - #22	N/A
🔴	☁	ICM Component Harness	8 mo 12 days - #730	12 days - #759
🔵	☀	ICM Component Harness Stable	2 yr 7 mo - #22	N/A
⚪	☀	ICM ECISDesktop Nightly (5.0.x)	1 mo 26 days - #379	N/A
🔵	☀	ICM ECISDesktop Nightly (Trunk)	12 days - #1665	N/A
🔵	☀	ICM ECISDesktop Regular (Trunk)	12 days - #742	N/A
🔵	☁	ICM Web Services 1.3 Regular (CXF)	5 mo 5 days - #458	5 mo 18 days - #455
🟡	⛅	ICM Web Services Prod ECIS 5.0	6 hr 41 min - #14	1 mo 11 days - #12
🔵	⛅	ICMServices Carbon Application	10 mo - #133	10 mo - #132