**Configuring Your Debugging Environment**

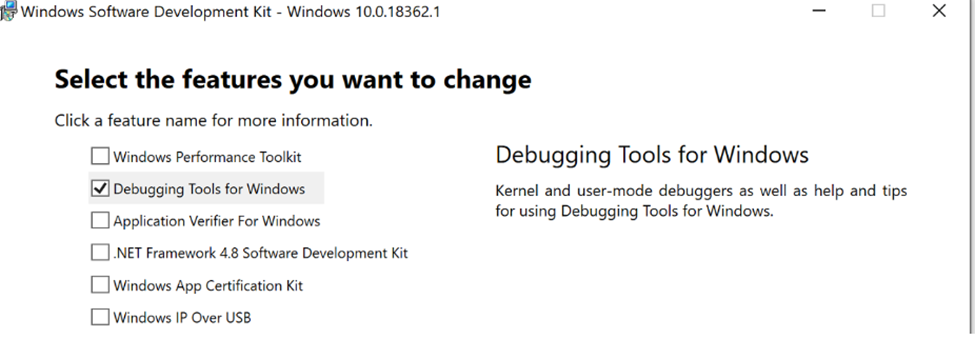
**Installing the Windows Debugger (WinDbg)**

The Debugging Tools for Windows (WinDbg) are included in Windows 10 SDK which can be downloaded here:

<https://developer.microsoft.com/en-us/windows/downloads/windows-10-sdk/>

If you just need the Debugging Tools for Windows, and not the Windows Driver Kit (WDK) for Windows 10, you can install the debugging tools as a standalone component.

In the SDK installation wizard, select Debugging Tools for Windows, and deselect all other components.



This will install two new applications:

"C:\Program Files (x86)\Windows Kits\10\Debuggers\x86\windbg.exe"

"C:\Program Files (x86)\Windows Kits\10\Debuggers\x64\windbg.exe"

Note: Use the version that matches the build of the app you are debugging

Client-Side DMPs – Use the 32-bit version

Server-Side DMPS – Use the 64-bit version

**CONFIGURE THE CMDTREE**

This will give you a list of commands available without needing to type manually.

1. Create a new txt file anywhere you can access called “commands.txt” However, it will be easier to save copies in the same paths as the windbg executables
2. Copy the text from the following link into the new file

<https://www.dumpanalysis.org/blog/files/CMDTREE.TXT>

Configuring a command tree is optional, but worth the time .You can also name the file anything you want as long as you remember what it is called.

**Configuring the Symbol Path in WinDBG**

When debugging, you must make sure that the debugger can access the symbol files that are associated with the target you are debugging. You must obtain the proper symbols for the code that you wish to debug and load these symbols into the debugger. The Symbol Path indicates where the debugger will find the needed symbols.

There are two locations needed when working with either client or server crashes within the eDOCS DM environment.

* Location of the eDOCS symbols for the version or client or server (ex. DM 10 Patch 5, 16.3..etc)
* The location of the Microsoft Public Symbol Server

(<https://msdl.microsoft.com/download/symbols>)

1. Start the Windows Debugger (WinDbg.exe).

2. On the File menu, click Symbol File Path.

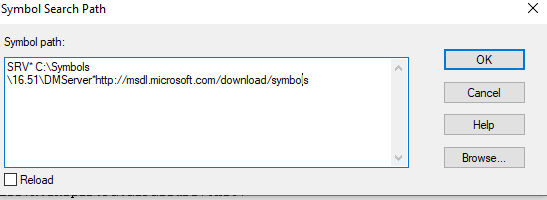
3. In the Symbol path box, type the following command:

SRV\*your local folder for symbols\*http://msdl.microsoft.com/download/symbols

“your local folder for symbols” is the location of the eDOCS DM symbols.

When you start to debug a crash dump file, the Windows Debugger checks whether the symbol information for a module that it loads is in the local folder. If the symbol information of the module is not located in the local folder, WinDbg tries to download the appropriate symbol file or files from the Symbol Server.

Example:



This path can also be entered, and verified via the .sympath command within the terminal once the dmp file is opened by running this command.

.sympath SRV\* C:\Symbols\16.51\DMServer\*http://msdl.microsoft.com/download/symbols

**HOW TO ANALYZE A DMP FILE WITH WinDbg:**

These steps assume you have already opened WinDbg.

1. Open a Crash DMP File

Choose Open Crash Dump from the File menu or by pressing CTRL+D. When the Open Crash Dump dialog box appears, enter the full path and name of the crash dump file in the File name box, or use the dialog box to select the proper path and file name. When the proper file has been chosen, click Open.

1. Load the CMD Tree

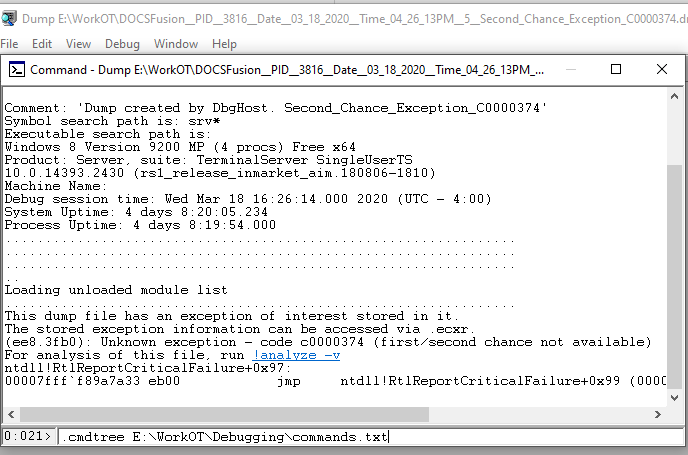
This step is optional if you know the commands you wish to run. However, performing this step will reduce the learning curve if just starting out in reading dmps

Invoke the CMD Tree window by running the following:

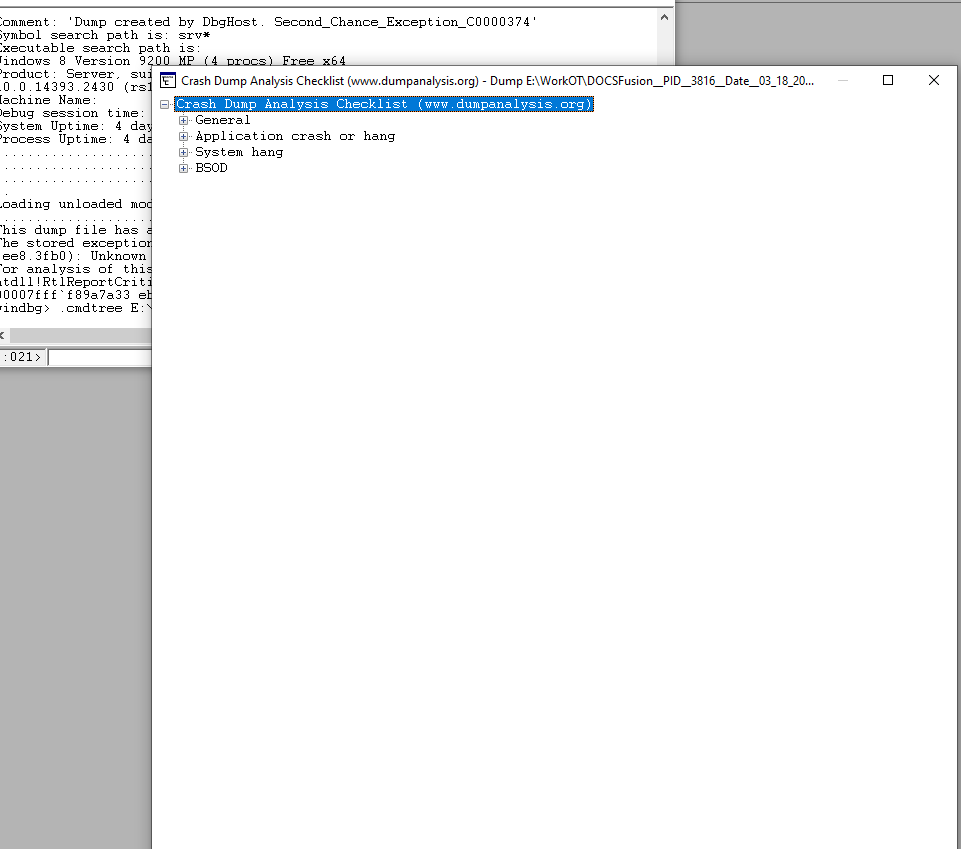
.cmdtree c:\commands.txt

Note: This the “commands.txt” file you created in the earlier part of this

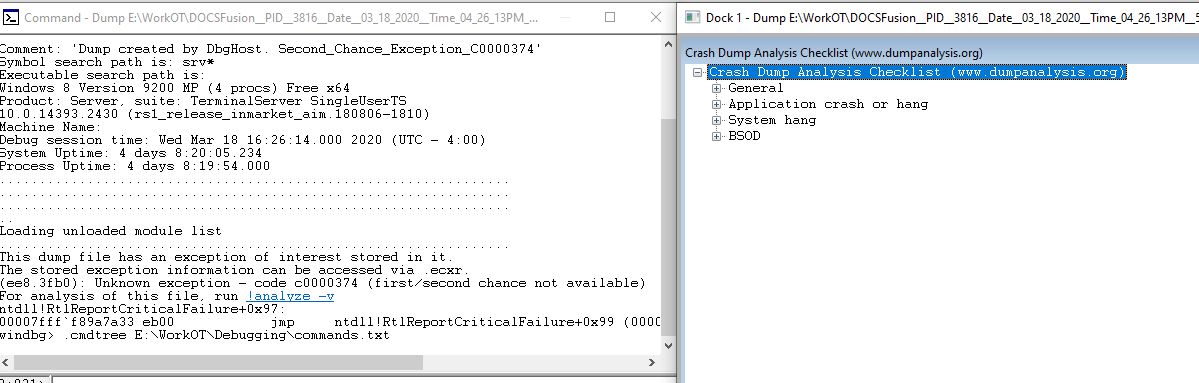
Ex.



You will see a new window appear. If you right click the top of the cmds window and choose “Move to new dock”, its easier to work with

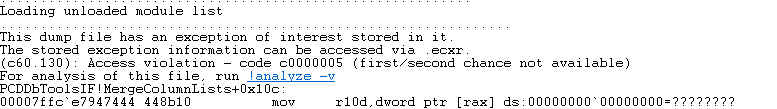


After pushing to a new doc pull the windows next to each other

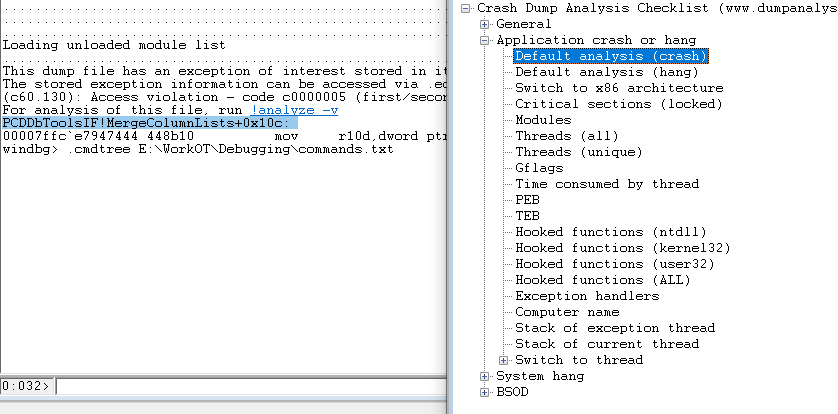


WinDbg is kind enough to tell you that there was an exception, and what command you can run to get more information:

The example below states an exception was thrown in PCDDbToolsIF!MergeColumnLists



You can simply type !analyze -v as directed, or choose Default Analysis(Crash)



This will produce a large report, you are mainly interested in the section labled Stack Text:

STACK\_TEXT:

00000048`8caff380 00007ffc`e3740094 : 00000229`dc1945b0 00000229`dc1944f0 00000000`00000000 00000000`00000000 : PCDDbToolsIF!MergeColumnLists+0x10c

00000048`8caff3d0 00007ffc`e794b8d5 : 00000229`dc1944f0 00007ffc`f9e29200 00007ffc`e7956900 00000229`ffffffff : PCDDbSearchStreamIF!CPCDSearchStreamFullTextJoining::StartFetch+0x7fc

00000048`8caff530 00007ffc`e373f881 : 00000229`dc1944f0 00000000`00000000 00000229`dc1944e8 00007ffc`f9e29216 : PCDDbToolsIF!CPCDSearchStreamBase::StartOnly+0x15d

00000048`8caff5b0 00007ffc`e628ea77 : 00000229`e7b817b0 00000229`dc1944e8 00000229`d571db60 00007ffc`e6f83d66 : PCDDbSearchStreamIF!CPCDSearchStreamFullTextJoining::Start+0x45

00000048`8caff5e0 00007ffc`e628bb50 : 00000229`d571db60 00000229`db19e358 00000000`00000000 00000229`db19e2a0 : PCDSearchStreamOperatorsIF!RowsetManager::LoadFromStream+0xaf

00000048`8caff6d0 00007ffc`e794b71c : 00000229`db19e2b0 00000229`d571db60 00007ffc`e7a39628 00000229`db19e380 : PCDSearchStreamOperatorsIF!CPCDSearchStreamSort::StartFetch+0x158

00000048`8caff750 00007ffc`e794b6e7 : 00000229`db19e2b0 00000000`0000000a 00000229`cafad520 00007ffc`e7a183f1 : PCDDbToolsIF!CPCDSearchStreamBase::StartInThread+0x1c

00000048`8caff7b0 00007ffc`e7a18642 : 00000048`8caff8d0 00000000`00000000 00007ffc`e7a3f050 00000000`000000ca : PCDDbToolsIF!CPCDSearchStreamBase::Start+0x197

00000048`8caff800 00007ffc`e7a1898a : 00000048`8caff8d0 00000000`00000000 0000a437`f5ea6073 00000000`00000000 : PCDut32!CPCDWorkerThread::Run+0xae

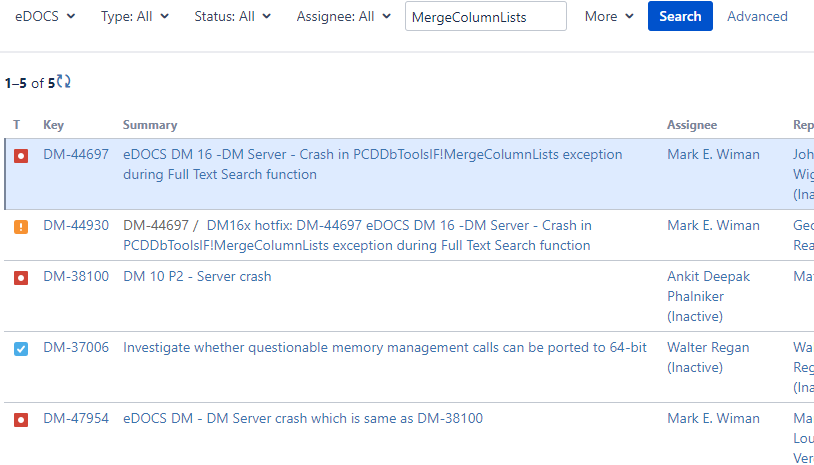
00000048`8caff890 00007ffc`f9e902c6 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : PCDut32!CPCDWorkerThread::StartThread+0x1ba

00000048`8caff8f0 00007ffc`fd5784d4 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : ucrtbase!thread\_start<void (\_\_cdecl\*)(void \* \_\_ptr64)>+0xa6

00000048`8caff920 00007ffc`fd7be851 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : kernel32!BaseThreadInitThunk+0x14

00000048`8caff950 00000000`00000000 : 00000000`00000000 00000000`00000000 00000000`00000000 00000000`00000000 : ntdll!RtlUserThreadStart+0x21

The next step is to use that information to search JIRA for known issues by the function name (MergeColumnLists)



You will have to look through the JIRAs for a matching stack , and possibly look at duplicates and subtasks to find a match. In this case the stack trace matches exactly to an older bug and in this case was a regression

