



# MAP File Browser

by

Software Verify

Copyright © 2015-2026 Software Verify Limited

# MAP File Browser

## MAP file contents inspector

---

*by Software Verify Limited*

*Welcome to the MAP File Browser software tool. MAP File Browser is a software tool that allows you to inspect the contents of MAP files.*

*We hope you will find this document useful.*

# MAP File Browser Help

**Copyright © 2015-2026 Software Verify Limited**

All rights reserved. No parts of this work may be reproduced in any form or by any means - graphic, electronic, or mechanical, including photocopying, recording, taping, or information storage and retrieval systems - without the written permission of the publisher.

Products that are referred to in this document may be either trademarks and/or registered trademarks of the respective owners. The publisher and the author make no claim to these trademarks.

While every precaution has been taken in the preparation of this document, the publisher and the author assume no responsibility for errors or omissions, or for damages resulting from the use of information contained in this document or from the use of programs and source code that may accompany it. In no event shall the publisher and the author be liable for any loss of profit or any other commercial damage caused or alleged to have been caused directly or indirectly by this document.

Printed: December 2025 in United Kingdom.

# Table of Contents

Foreword	1
<b>Part I How to get MapFileBrowser</b>	<b>2</b>
<b>Part II What does MapFileBrowser do?</b>	<b>4</b>
<b>Part III What is a module?</b>	<b>6</b>
<b>Part IV MAP File contents</b>	<b>8</b>
<b>Part V Menu</b>	<b>13</b>
1 File .....	14
2 Settings .....	14
3 Query .....	14
4 Software Updates .....	15
5 Help .....	18
<b>Part VI The user interface</b>	<b>21</b>
<b>Part VII Settings dialog</b>	<b>25</b>
1 Symbols .....	26
2 Data display .....	27
3 Source code paths .....	28
4 Path Substitutions .....	30
<b>Part VIII MAP File Information</b>	<b>32</b>
<b>Part IX How to use MapFileBrowser</b>	<b>34</b>
1 Decoding an absolute crash address .....	36
2 Decoding a relative crash address .....	38
3 Decoding a symbol relative crash address .....	41
4 Decoding an Event Viewer XML crash log .....	43
5 What is a load address? .....	46
<b>Part X Command Line Interface</b>	<b>54</b>
<b>Index</b>	<b>0</b>



**Part**



# 1 How to get MapFileBrowser

MapFileBrowser is free for commercial use. MapFileBrowser can be downloaded for Software Verify's website at <https://www.softwareverify.com/product/map-file-browser/>.

This help manual is available in Compiled HTML Help (Windows Help files), PDF, and online.

Windows Help	<a href="https://www.softwareverify.com/documentation/chm/mapFileBrowser.chm">https://www.softwareverify.com/documentation/chm/mapFileBrowser.chm</a>
PDF	<a href="https://www.softwareverify.com/documentation/pdfs/mapFileBrowser.pdf">https://www.softwareverify.com/documentation/pdfs/mapFileBrowser.pdf</a>
Online	<a href="https://www.softwareverify.com/documentation/html/mapFileBrowser/index.html">https://www.softwareverify.com/documentation/html/mapFileBrowser/index.html</a>

Whilst MapFileBrowser is free for commercial use, MapFileBrowser is copyrighted software and is not in the public domain.

You are free to use the software at your own risk.

You are not allowed to distribute the software in any form, or to sell the software, or to host the software on a website.

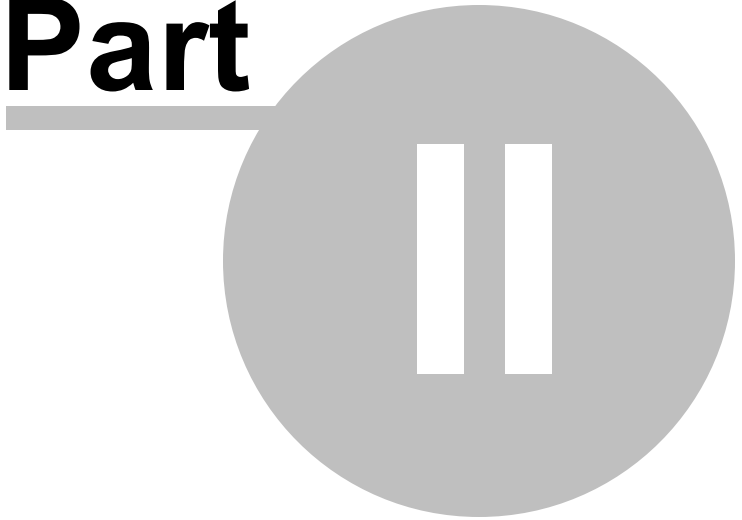
Contact Software Verify at:

Software Verify Limited  
Suffolk Business Park  
Eldo House  
Kempson Way  
Bury Saint Edmunds  
IP32 7AR  
United Kingdom

email [sales@softwareverify.com](mailto:sales@softwareverify.com)  
web <https://www.softwareverify.com>  
blog <https://www.softwareverify.com/blog>  
twitter <http://twitter.com/softwareverify>

Visit our blog to read our articles on debugging techniques and tools.  
Follow us on twitter to keep track of the latest software tools and updates.

**Part**



## 2 What does MapFileBrowser do?

MapFileBrowser allows you to inspect the contents of a linker MAP file.

You can sort the data, filter the data by name.

You can also query the data by address which can be useful for identifying what function is at a given address if all you have is a crash address and nothing else.

Query by address is supported three ways:

- Query by absolute address.
- Query by address offset from a DLL load address.
- Query by address offset from a symbol.

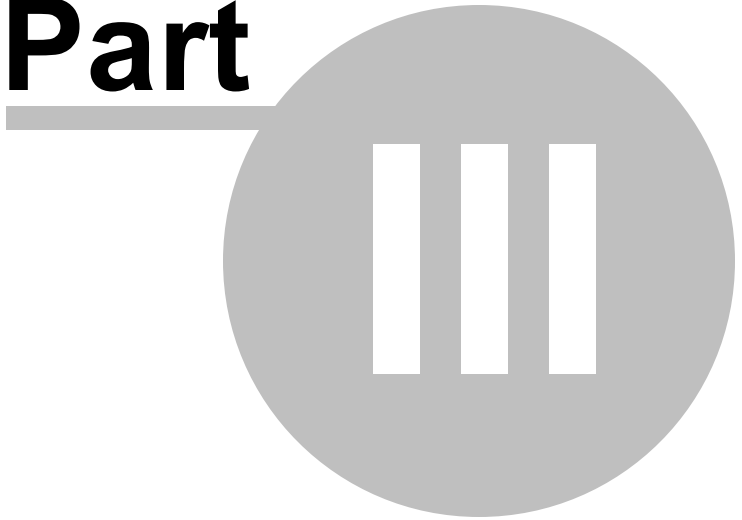
### 32 bit and 64 bit

MAP files created by 32 bit and 64 bit software are supported. On 64 bit Operating systems if a 64 bit MAP file is opened the 64 bit version MAP File Browser is automatically started.

### History

MapFileBrowser has been an internal tool at Software Verify for many years. We recently decided to make it a bit more user friendly and to make it available for public use.

**Part**



### 3 What is a module?

A module is a contained block of executable code and data. For example, a DLL or EXE.

Some software vendors name their DLLs with different file extensions, for example .BPL, .ARX.

When you call `LoadLibrary` to load a module, you are returned a `HMODULE`, which is an opaque handle to a module. The `HMODULE` is most often the same as the module load address, but not always. The lower few bits of the `HMODULE` can get OR'd with some flags to create a `HMODULE` value that is not the same as the module load address.

You can get the load address of a module from its `HMODULE` by masking out the lower 16 bits of the `HMODULE` value then casting to a `DWORD_PTR`.

In this documentation when you read EXE or DLL or module, we are effectively referring to the same thing. It's easier to read and write "DLLs" rather than "DLLs or EXE".

**Part**



## 4 MAP File contents

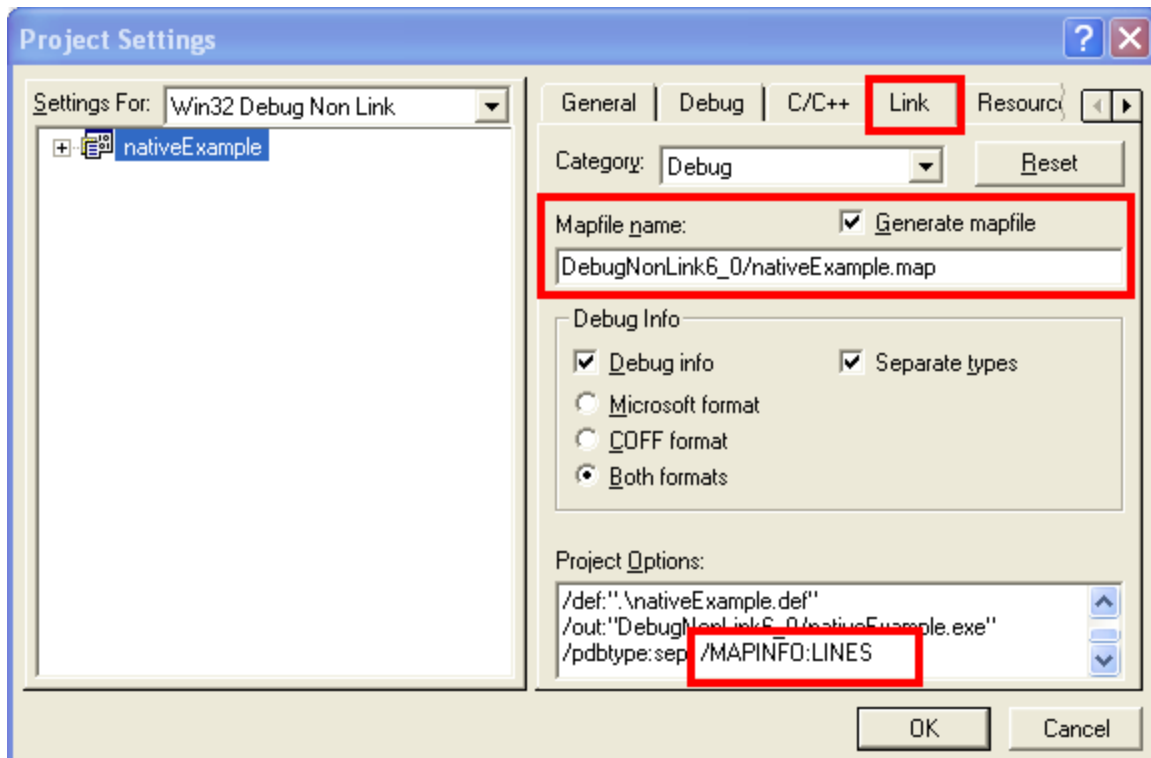
MapFileBrowser can read MAP files generated by Microsoft Visual Studio, Delphi, C++ Builder and various other MAP files.

Some MAP files contain line numbers, some don't. The options for adding line numbers to MAP files have changed with different versions of compilers and linkers.

Even if a MAP file doesn't contain line numbers, the MAP can still be useful in guiding you to the correct function that is related to an address.

### Visual Studio 6

Visual Studio 6 will add line numbers to your MAP file if you add the following option to your linker settings /MAPINFO:LINES

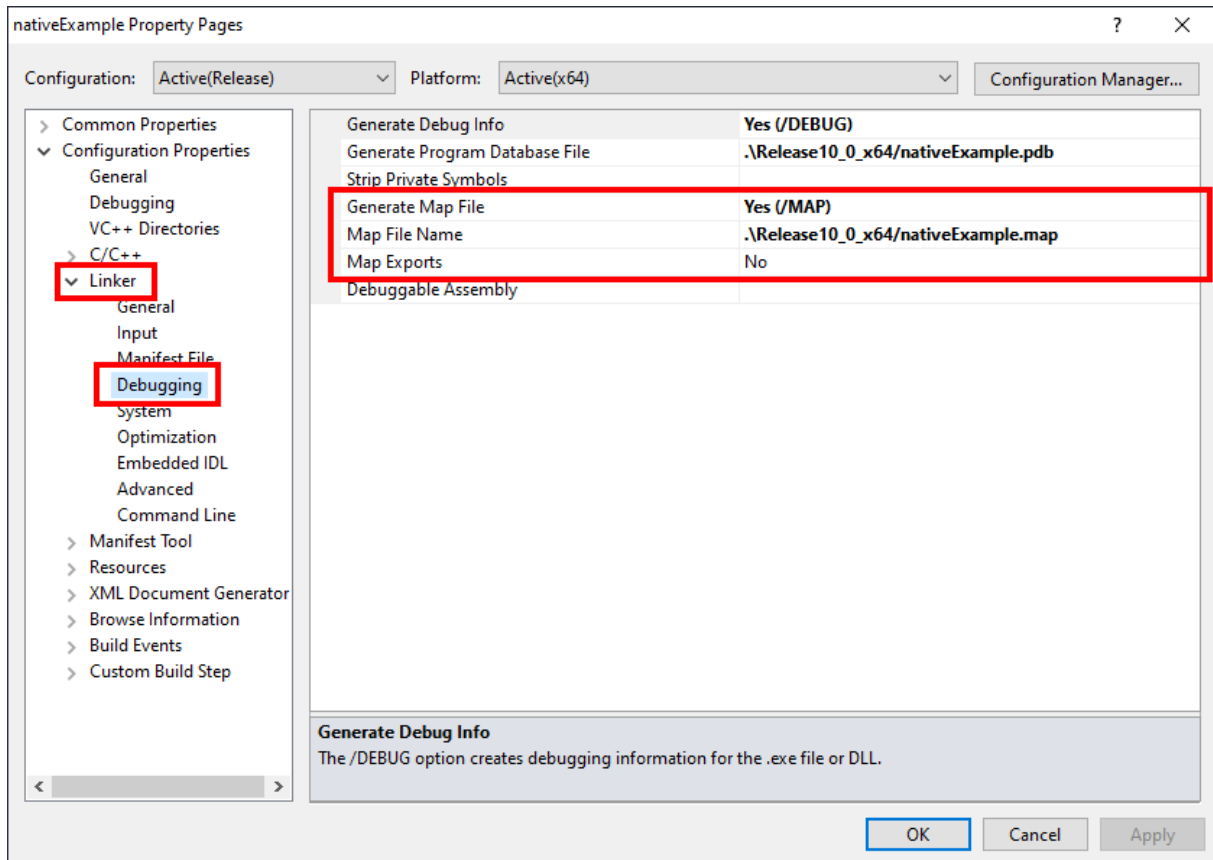


### Visual Studio 7 (2002) and later

Visual Studio 7 and later will not add line numbers to your MAP file. The linker option /MAPINFO:LINES is not supported.

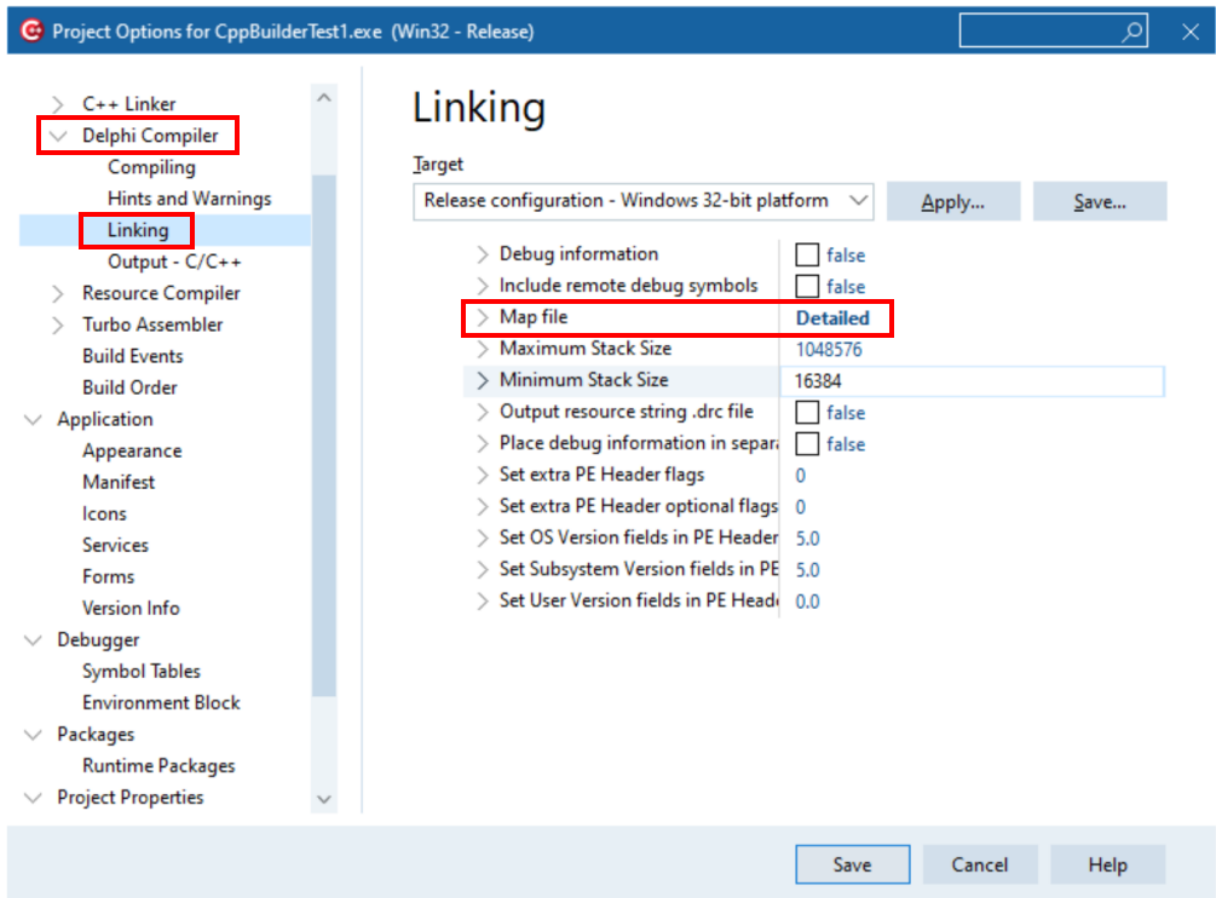
This means you can't create MAP files containing line numbers for anything you build with Visual Studio 7 (2002) or later, or for any 64 bit builds.

You can still use MapFileBrowser to identify the function which contains a given address.



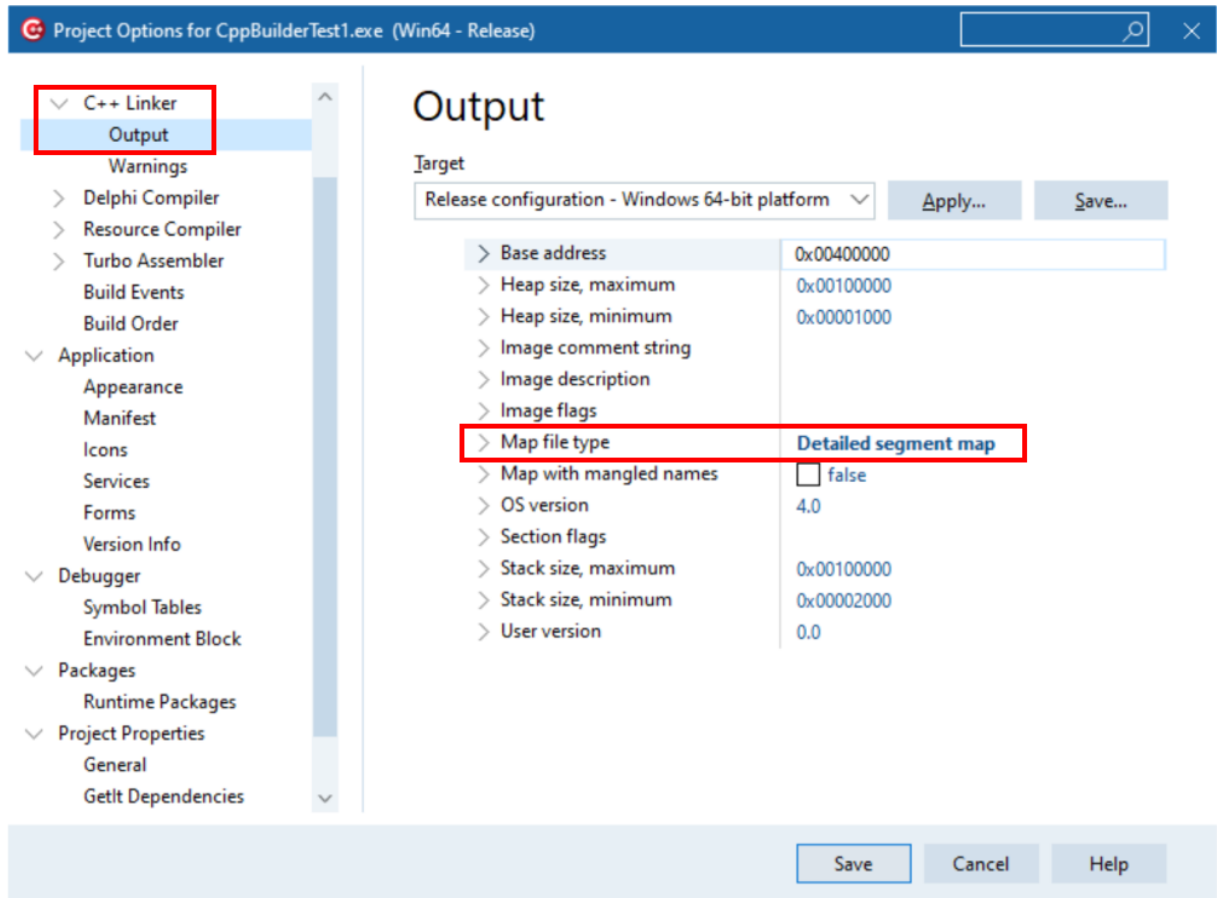
## Delphi

To create MAP files containing line numbers you need to specify that you want a "detailed" MAP file in the options for your Delphi or C++ Builder project.



## C++ Builder

To create MAP files containing line numbers you need to specify that you want a "detailed" MAP file in the options for your Delphi or C++ Builder project.

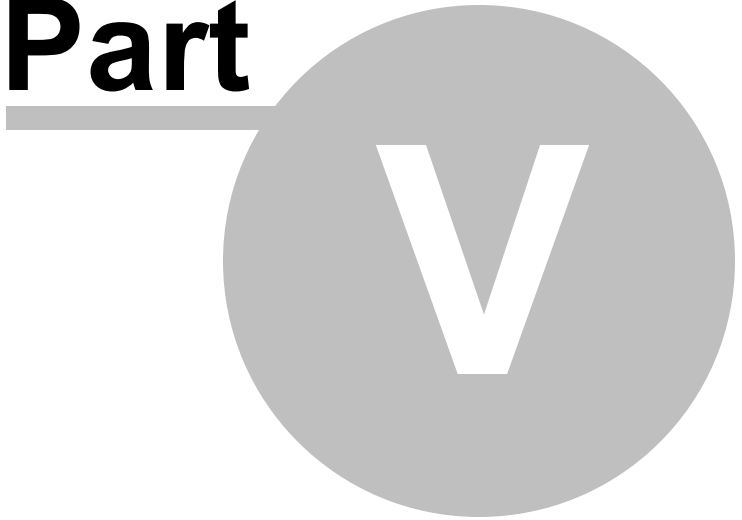


### Other Compilers

If you find that MapFileBrowser cannot read your MAP file, please contact us at [support@softwareverify.com](mailto:support@softwareverify.com), attaching your MAP file to the email.

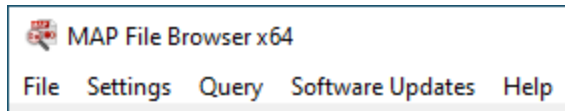
We will try to add support for your MAP file format as soon as we can.

**Part**



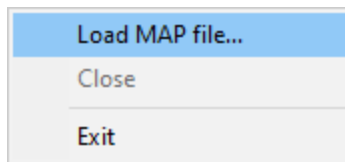
## 5 Menu

The main menu contains five menus, File, Edit, Query, Software Updates and Help.



### 5.1 File

The File menu controls loading of MAP file information, clearing the display and exiting the program.



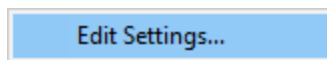
**File** menu > **Load MAP file...** > loads a MAP file and displays it.

**File** menu > **Close** > clear all results, unloads the MAP file information.

**File** menu > **Exit** > closes MapFileBrowser.

### 5.2 Settings

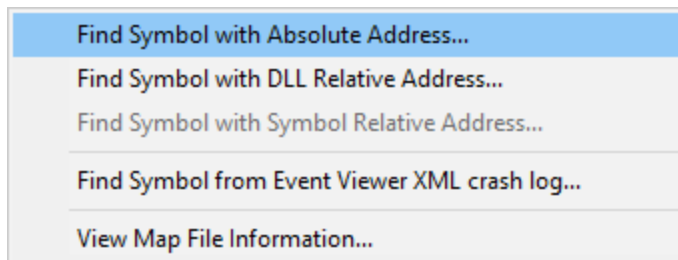
The Settings menu controls editing settings.



**Settings** menu > **Edit Settings...** > displays the settings dialog.

### 5.3 Query

The Query menu controls searching for symbols.



**Query menu > Find Symbol with Absolute Address...** > Use this option to turn an absolute address in a process into a symbol, filename and line number.

See Decoding an absolute crash address for more details.

**Query menu > Find Symbol with DLL Relative Address...** > use this option to turn a relative address inside a DLL into a symbol, filename and line number.

See Decoding a relative crash address for more details.

**Query menu > Find Symbol with Symbol Relative Address...** > use this option to turn a address that is relative to a symbol inside a DLL into a symbol, filename and line number.

See Decoding a symbol relative crash address for more details.

**Query menu > Find Symbol from Event Viewer XML crash log...** > use this option to turn an XML crash log from the Microsoft Event Viewer to a symbol inside a DLL into a symbol, filename and line number.

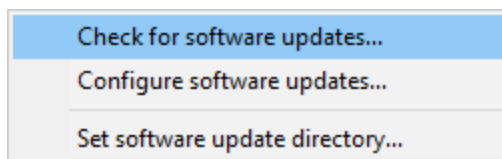
See Decoding an Event Viewer XML crash log for more details.

**Query menu > View Map File Information...** > use this option to view information about the MAP file.

## 5.4 Software Updates


The Software Updates menu controls how often software updates are downloaded.

If you've been notified of a new software release to Map File Browser or just want to see if there's a new version, this feature makes it easy to update.



 **Software Updates menu > Check for software updates** > checks for updates and shows the software update dialog if any exist

An internet connection is needed to be able to make contact with our servers.

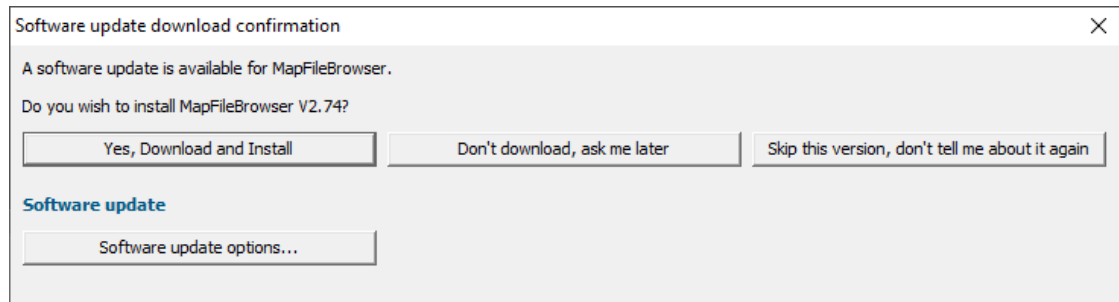
 Before updating the software, close the help manual, and end any active session by closing target programs.

If no updates are available, you'll just see this message:

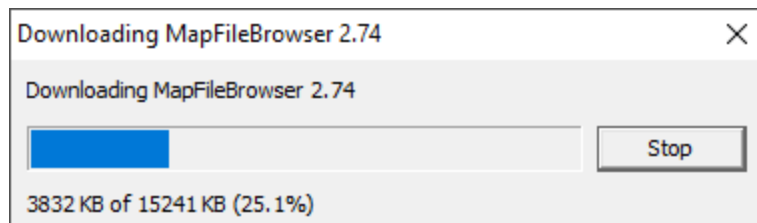


## Software Update dialog

If a software update is available for Map File Browser you'll see the software update dialog.



- **Download and install** > downloads the update, showing progress



Once the update has downloaded, Map File Browser will close, run the installer, and restart.

You can stop the download at any time, if necessary.

- **Don't download...** > Doesn't download, but you'll be prompted for it again next time you start Map File Browser
- **Skip this version...** > Doesn't download the update and doesn't bother you again until there's an even newer update
- **Software update options...** > edit the software update schedule

## Problems downloading or installing?

If for whatever reason, automatic download and installation fails to complete:

- Download the latest installer manually from the software verify website.

Make some checks for possible scenarios where files may be locked by Map File Browser as follows:

- Ensure Map File Browser and its help manual is also closed
- Ensure any error dialogs from the previous installation are closed

You should now be ready to run the new version.

## Software update schedule

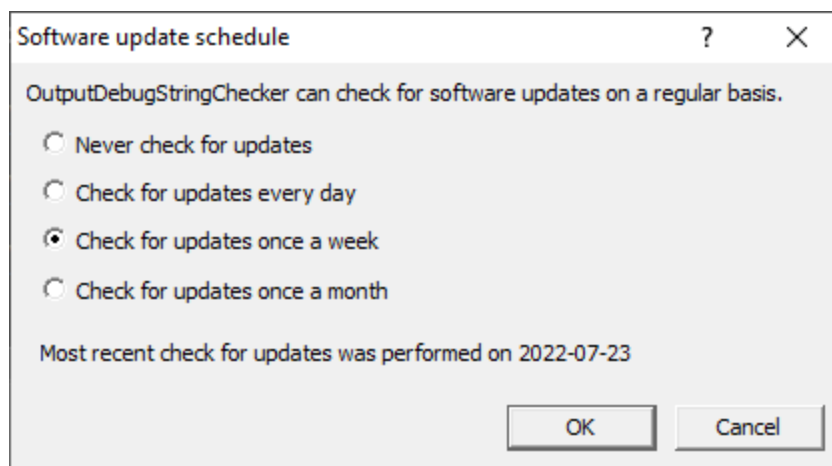
Map File Browser can automatically check to see if a new version of Map File Browser is available for downloading.

 **Software Updates** menu > **Configure software updates** > shows the software update schedule dialog

The update options are:

- never check for updates
- check daily (the default)
- check weekly
- check monthly

The most recent check for updates is shown at the bottom.

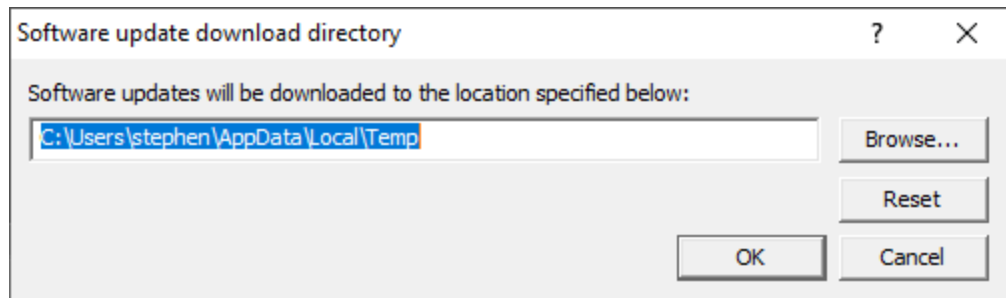


## Software update directory

It's important to be able to specify where software updates are downloaded to because of potential security risks that may arise from allowing the `TMP` directory to be executable. For example, to counteract security threats it's possible that account ownership permissions or antivirus software blocks program execution directly from the `TMP` directory.

The `TMP` directory is the default location but if for whatever reason you're not comfortable with that, you can specify your preferred download directory. This allows you to set permissions for `TMP` to deny execute privileges if you wish.


 **Software Updates** menu > **Set software update directory** > shows the Software update download directory dialog



An invalid directory will show the path in red and will not be accepted until a valid folder is entered.

Example reasons for invalid directories include:

- the directory doesn't exist
- the directory doesn't have write privilege (update can't be downloaded)
- the directory doesn't have execute privilege (downloaded update can't be run)

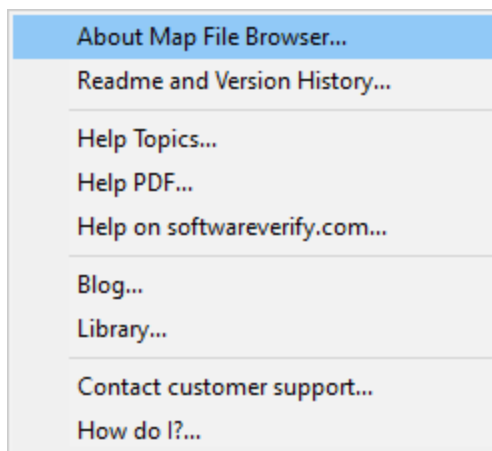
 When modifying the download directory, you should ensure the directory will continue to be valid. Updates may no longer occur if the download location is later invalidated.

- **Reset** > reverts the download location to the user's `TEMP` directory

The default location is `c:\users\[username]\AppData\Local\Temp`

## 5.5 Help

The Help menu controls displaying this help document and displaying information about Map File Browser.



**Help menu** > **About Map File Browser...** > displays information about Map File Browser.

**Help menu** > **Readme and Version History...** > displays the readme and version history.

**Help menu** > **Help Topics...** > displays this help file.

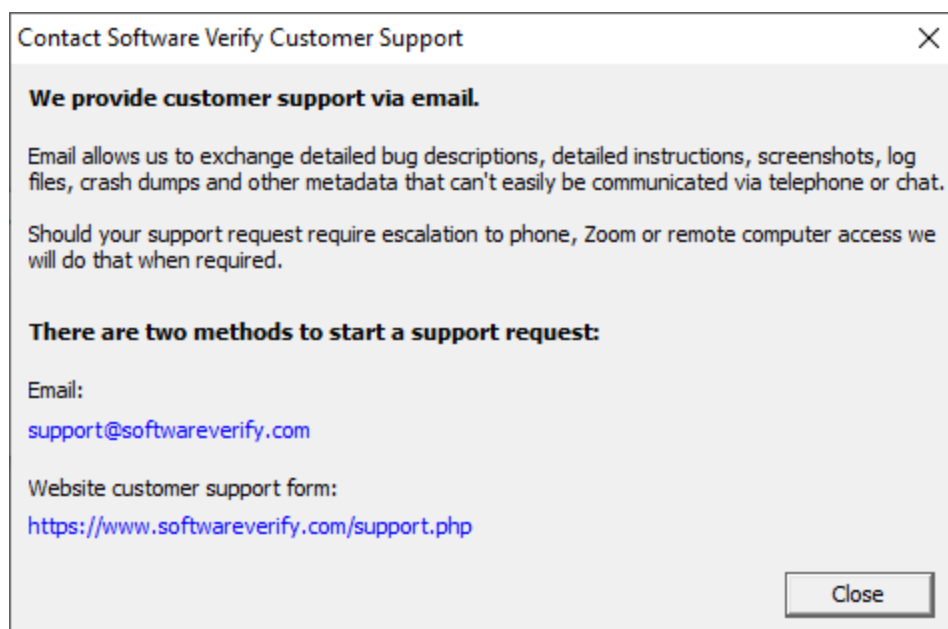
**Help menu** > **Help PDF...** > displays this help file in PDF format.

**Help menu** > **Help on softwareverify.com...** > display the Software Verify documentation web page where you can view online documentation or download compiled HTML Help and PDF help documents.

**Help menu** > **Blog...** > display the Software Verify blog.

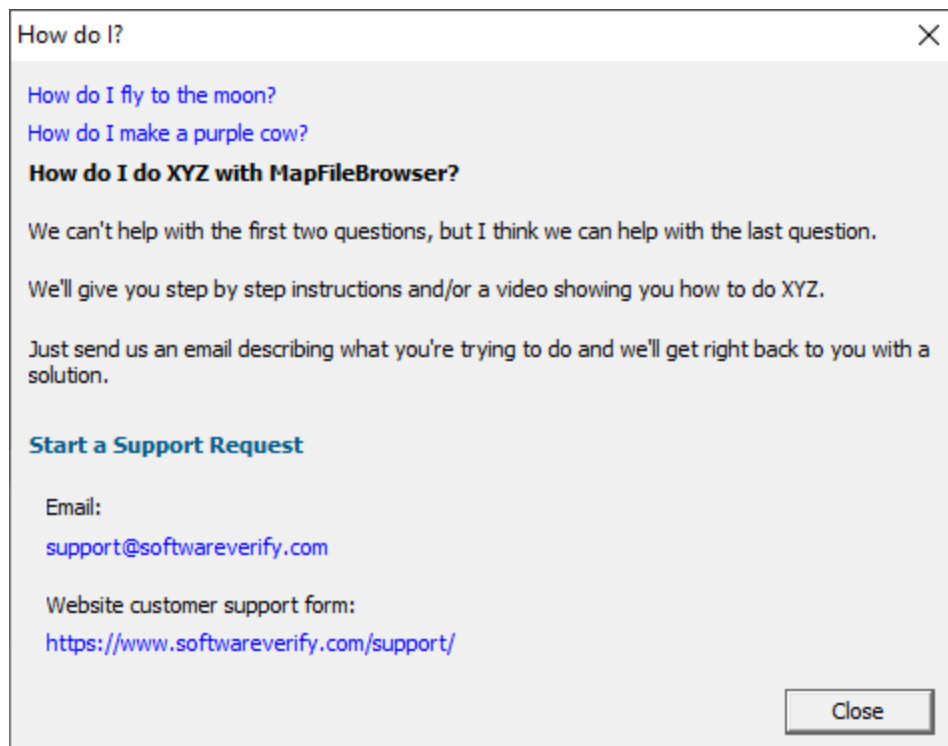
**Help menu** > **Library...** > display the Software Verify library - our best blog articles grouped by related topics.

**Help menu** > **Contact customer support...** > displays the options for contacting customer support.



Click a link to contact customer support.

**Help menu** > **How do I?...** > displays the options for asking us how to do a particular task.



**Part**



## 6 The user interface

The MapFileBrowser user interface is shown below.

The screenshot shows the MAP File Browser application window. The main window displays a table of functions with columns for Line #, Name (634), Segment, Section, Address, Size, Type, Filename, and Line. The function at line 33, `public __thiscall svDataTracker::svDataTracker(char const *)`, is selected. Below the table, the 'Line Numbers' and 'Filename' sections show the source code for this function, including comments and the function definition.

#	Name (634)	Segment	Section	Address	Size	Type	Filename	Line
30	public __thiscall exampleIncrementDecrement::exampleIncrementDecrement(void)	0x0001	.text	0x0040A0A0	64	Public	E:\om\c\memory32\examples\nativeExample\TESTSVW.CPP	227
31	public __thiscall heapNewBaseClass::heapNewBaseClass(void)	0x0001	.text	0x00401040	64	Public	E:\om\c\memory32\examples\nativeExample\heapNewBaseClass.cpp	72
32	public __thiscall std::_Tree<unsigned long,struct std::pair<unsigned long, unsigned lo...	0x0001	.text	0x004299E0	96	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\INCLUDE\xtree	122
33	public __thiscall svDataTracker::svDataTracker(char const *)	0x0001	.text	0x0042A700	96	Public	E:\om\c\memory32\API\svDataTracker.cpp	17
34	public __thiscall svDataTracker::svDataTracker(unsigned short const *)	0x0001	.text	0x0042A760	96	Public	E:\om\c\memory32\API\svDataTracker.cpp	33
35	public __thiscall testClass1::testClass1(unsigned long)	0x0001	.text	0x00402A80	64	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.h	30
36	public __thiscall testClass2::testClass2(unsigned short)	0x0001	.text	0x00402AC0	96	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.h	42
37	public __thiscall testClass3::testClass3(void)	0x0001	.text	0x00402B20	64	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.h	54
38	public __thiscall testClassToBeThrown::testClassToBeThrown(class testClassToBeThro...	0x0001	.text	0x00402120	192	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.CPP	149
39	public __thiscall testClassToBeThrown::testClassToBeThrown(void)	0x0001	.text	0x004020A0	128	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.CPP	142
40	private __thiscall type_info::type_info(class type_info const &)	0x0001	.text	0x0042E660	64	Public	ti_inst.cpp	17
41	public __thiscall std::_Tree<unsigned long,struct std::pair<unsigned long, unsigned lo...	0x0001	.text	0x00427D80	480	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\INCLUDE\xtree	174
42	public __thiscall ATL::CComBSTR::CComBSTR(void)	0x0001	.text	0x00421820	48	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\ATL\INCLUDE\atlba...	3943
43	public __thiscall ATL::CComVariant::CComVariant(void)	0x0001	.text	0x00421820	64	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\ATL\INCLUDE\atlba...	4175
44	public virtual __thiscall CMainFrame::CMainFrame(void)	0x0001	.text	0x00401880	96	Public	E:\om\c\memory32\examples\nativeExample\MAINFRM.CPP	56

Line Numbers: 17 18 19  
 Address: 0x0042A700 0x0042A70E 0x0042A71A  
 Offset: 0 14 26  
 Filename: E:\om\c\memory32\API\svDataTracker.cpp Line 17  
 Function: public \_\_thiscall svDataTracker::svDataTracker(char const \*)  
 Address: 0x0042A700

```

9 // when this object is deleted the tag tracker will be automatically popped from the stack of tag trackers.
10 // The top most tag tracker is used to tag allocations.
11 // PARAMETERS:
12 //   trackerName --in-- Name of tag tracker
13 // RETURN CODES:
14 // -----
15
16 svDataTracker::svDataTracker(const char *trackerName)
17 {
18     #ifdef TRACKER
19         #ifdef TRACKER_PUSH
20             #ifdef TRACKER_PUSH_TRACKER
21             #endif
22             #endif
23             #endif
24             #endif
25             #endif
26             #endif
27             #endif
28             #endif
29             #endif
30             #endif
31             #endif
32             #endif
33             #endif
34             #endif
35             #endif
36             #endif
37             #endif
38             #endif
39             #endif
40             #endif
41             #endif
42             #endif
43             #endif
44             #endif
45             #endif
46             #endif
47             #endif
48             #endif
49             #endif
50             #endif
51             #endif
52             #endif
53             #endif
54             #endif
55             #endif
56             #endif
57             #endif
58             #endif
59             #endif
60             #endif
61             #endif
62             #endif
63             #endif
64             #endif
65             #endif
66             #endif
67             #endif
68             #endif
69             #endif
70             #endif
71             #endif
72             #endif
73             #endif
74             #endif
75             #endif
76             #endif
77             #endif
78             #endif
79             #endif
80             #endif
81             #endif
82             #endif
83             #endif
84             #endif
85             #endif
86             #endif
87             #endif
88             #endif
89             #endif
90             #endif
91             #endif
92             #endif
93             #endif
94             #endif
95             #endif
96             #endif
97             #endif
98             #endif
99             #endif
100            #endif

```

The user interface consists of a main grid showing functions in the MAP file.

Below is a display for line numbers and a source code display for viewing the source code of any function or variable that is selected.

Selecting any item in the grid populates the line numbers and source code display as appropriate.

Querying any value will select the nearest item in the main grid and populate the other displays as appropriate.

Some basic filtering functionality is also provided.

### MAP File Information

#	Name (634)	Segment	Section	Address	Size	Type	Filename	Line
30	public __thiscall exampleIncrementDecrement::exampleIncrementDecrement(void)	0x0001	.text	0x0040A0A0	64	Public	E:\om\c\memory32\examples\nativeExample\TESTSVW.CPP	227
31	public __thiscall heapNewBaseClass::heapNewBaseClass(void)	0x0001	.text	0x00401040	64	Public	E:\om\c\memory32\examples\nativeExample\heapNewBaseClass.cpp	72
32	public __thiscall std::_Tree<unsigned long,struct std::pair<unsigned long, unsigned lo...	0x0001	.text	0x004299E0	96	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\INCLUDE\xtree	122
33	public __thiscall svDataTracker::svDataTracker(char const *)	0x0001	.text	0x0042A700	96	Public	E:\om\c\memory32\API\svDataTracker.cpp	17
34	public __thiscall svDataTracker::svDataTracker(unsigned short const *)	0x0001	.text	0x0042A760	96	Public	E:\om\c\memory32\API\svDataTracker.cpp	33
35	public __thiscall testClass1::testClass1(unsigned long)	0x0001	.text	0x00402A80	64	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.h	30
36	public __thiscall testClass2::testClass2(unsigned short)	0x0001	.text	0x00402AC0	96	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.h	42
37	public __thiscall testClass3::testClass3(void)	0x0001	.text	0x00402B20	64	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.h	54
38	public __thiscall testClassToBeThrown::testClassToBeThrown(class testClassToBeThro...	0x0001	.text	0x00402120	192	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.CPP	149
39	public __thiscall testClassToBeThrown::testClassToBeThrown(void)	0x0001	.text	0x004020A0	128	Public	E:\om\c\memory32\examples\nativeExample\nativeExample.CPP	142
40	private __thiscall type_info::type_info(class type_info const &)	0x0001	.text	0x0042E660	64	Public	ti_inst.cpp	17
41	public __thiscall std::_Tree<unsigned long,struct std::pair<unsigned long, unsigned lo...	0x0001	.text	0x00427D80	480	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\INCLUDE\xtree	174
42	public __thiscall ATL::CComBSTR::CComBSTR(void)	0x0001	.text	0x00421820	48	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\ATL\INCLUDE\atlba...	3943
43	public __thiscall ATL::CComVariant::CComVariant(void)	0x0001	.text	0x00421820	64	Public	C:\Program Files (x86)\Microsoft Visual Studio\VC98\ATL\INCLUDE\atlba...	4175
44	public virtual __thiscall CMainFrame::CMainFrame(void)	0x0001	.text	0x00401880	96	Public	E:\om\c\memory32\examples\nativeExample\MAINFRM.CPP	56

The MAP File information shows you the symbol name, segment number, segment name, symbol address, symbol size, symbol type, and the filename and line number for the symbol.

You can sort the data by clicking on the column header and clicking again to reverse the direction of the sort.

If you select any item in the grid the lower grids and source code display are populated with data as appropriate.

If you right click any item a context is displayed which will allow you to perform a symbol relative query.

cvExample.exe:Fri Jul 03 10:09:24 2020

## Line Numbers

Line #...	Address	Offset
530	0x00404D60	0
536	0x00404D7F	31
537	0x00404D96	54
545	0x00404D9C	60
546	0x00404DB7	87
548	0x00404DBD	93
551	0x00404DD0	112

The line numbers section lists each line number, the address of that line and the offset of that line from the start of the owning function. Note that offsets can be negative as well as positive depending on how the compiler did it's work.

## Source Code

```

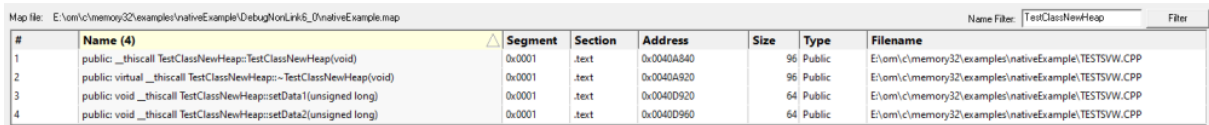
Filename: E:\om\c\memory32\mv\Example\..stubb\stubb.h Line 530
Function: public: __thiscall svDataTracker::svDataTracker(char *)
Address: 0x00404D60
Copy
522 // svDataTracker class, only available if using C++, thus handled by a separate ifdef
523 // see help for details on usage
524
525 #ifdef __cplusplus
526
527 class svDataTracker
528 {
529 public:
530 svDataTracker(char *trackerName)
531 {
532     HMODULE hMod;
533
534     // get module, will only succeed if Memory Validator launched this app or is injected into this app
535
536     hMod = GetModuleHandle(INJECTED_DLL);
537     if (hMod != NULL)
538     {
539         // MV is present, lookup the function and call it to push a new tracker onto the tracker stack for this thread
540     }
541 }
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681
2682
2683
2684
2685
2686
2687
2688
2689
2690
2691
2692
2693
2694
2695
2696
2697
2698
2699
2700
2701
2702
2703
2704
2705
2706
2707
2708
2709
2710
2711
2712
2713
2714
2715
2716
2717
2718
2719
2720
2721
2722
2723
2724
2725
2726
2727
2728
2729
2730
2731
2732
2733
2734
2735
2736
2737
2738
2739
2740
2741
2742
2743
2744
2745
2746
2747
2748
2749
2750
2751
2752
2753
2754
2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784
2785
2786
2787
2788
2789
2790
2791
2792
2793
2794
2795
2796
2797
2798
2799
2800
2801
2802
2803
2804
2805
2806
2807
2808
2809
2810
2811
2812
2813
2814
2815
2816
2817
2818
2819
2820
2821
2822
2823
2824
2825
2826
2827
2828
2829
2830
2831
2832
2833
2834
2835
2836
2837
2838
2839
2840
2841
2842
2843
2844
2845
2846
2847
2848
2849
2850
2851
2852
2853
2854
2855
2856
2857
2858
2859
2860
2861
2862
2863
2864
2865
2866
2867
2868
2869
2870
2871

```

The filters section allows you to filter data by symbol name.

### Name Filter

Filtering by symbol name allows you to easily find a particular symbol. This is very useful when wanting to decode a crash address that has been provided as relative to a symbol (symbol + offset).

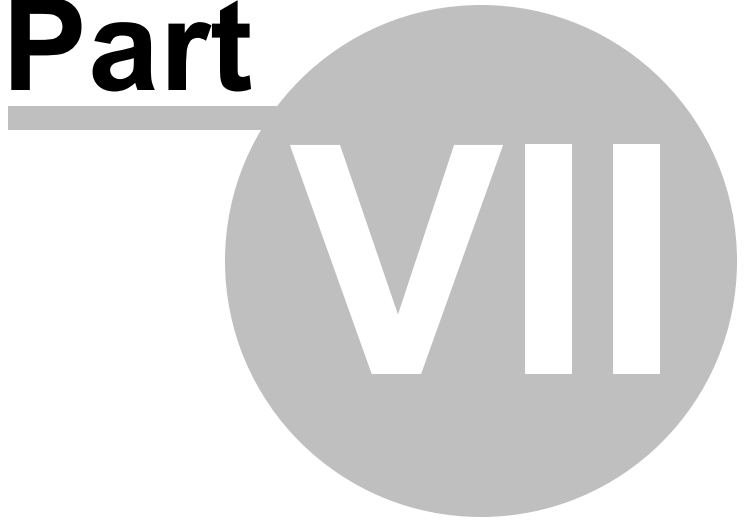


The screenshot shows a MAP File Browser window with the following details:

- Map file: E:\om\c\memory32\examples\nativeExample\Debug\omLink6\_0\nativeExample.map
- Name Filter: TestClassNewHeap
- Filter: (empty)

#	Name (4)	Segment	Section	Address	Size	Type	Filename
1	public: __thiscall TestClassNewHeap::TestClassNewHeap(void)	0x0001	.text	0x0040A840	96	Public	E:\om\c\memory32\examples\nativeExample\TESTSVW.CPP
2	public: virtual __thiscall TestClassNewHeap::~TestClassNewHeap(void)	0x0001	.text	0x0040A920	96	Public	E:\om\c\memory32\examples\nativeExample\TESTSVW.CPP
3	public: void __thiscall TestClassNewHeap::setData1(unsigned long)	0x0001	.text	0x0040D920	64	Public	E:\om\c\memory32\examples\nativeExample\TESTSVW.CPP
4	public: void __thiscall TestClassNewHeap::setData2(unsigned long)	0x0001	.text	0x0040D960	64	Public	E:\om\c\memory32\examples\nativeExample\TESTSVW.CPP

**Part**

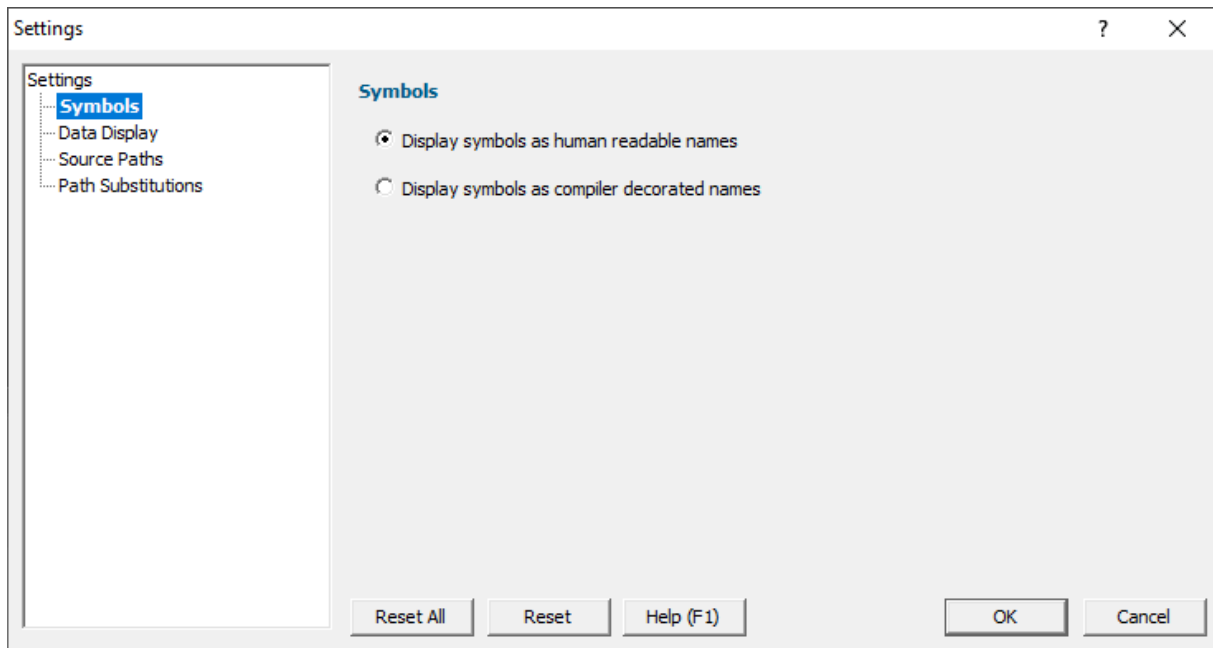


**VII**

## 7 Settings dialog

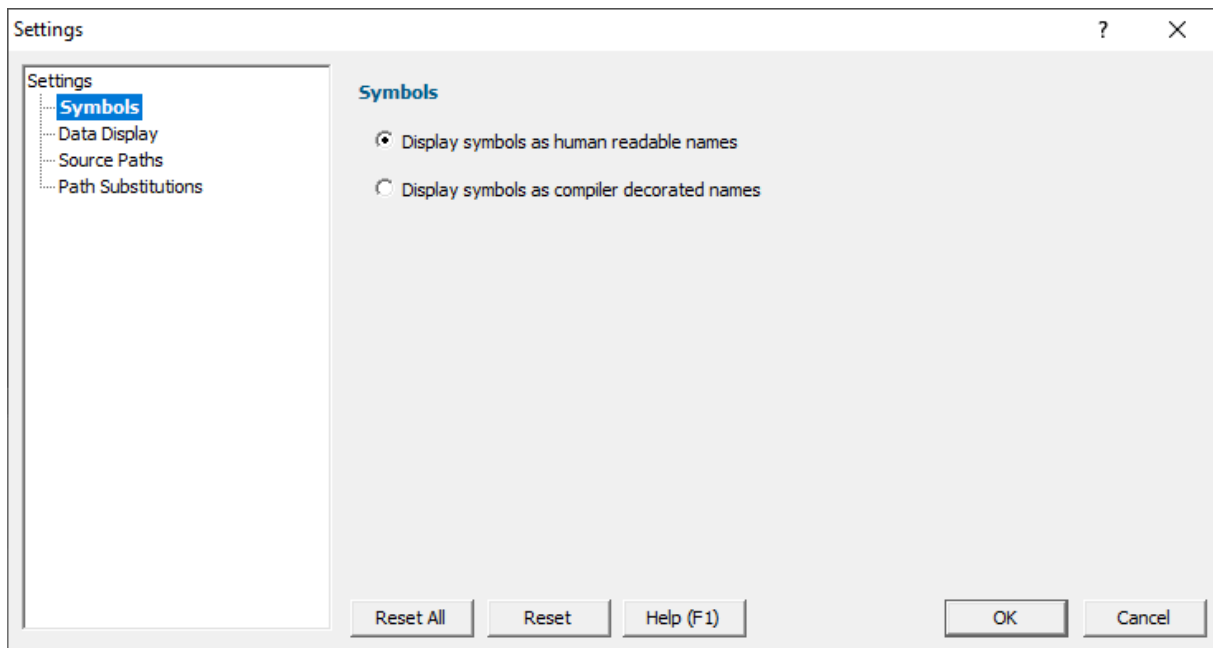
The settings dialog allows you to control the behaviour of Map File Browser.

There are three main sections: Symbols, Data display and source code paths.



### 7.1 Symbols

The Symbols settings allow you to choose how symbols are displayed.



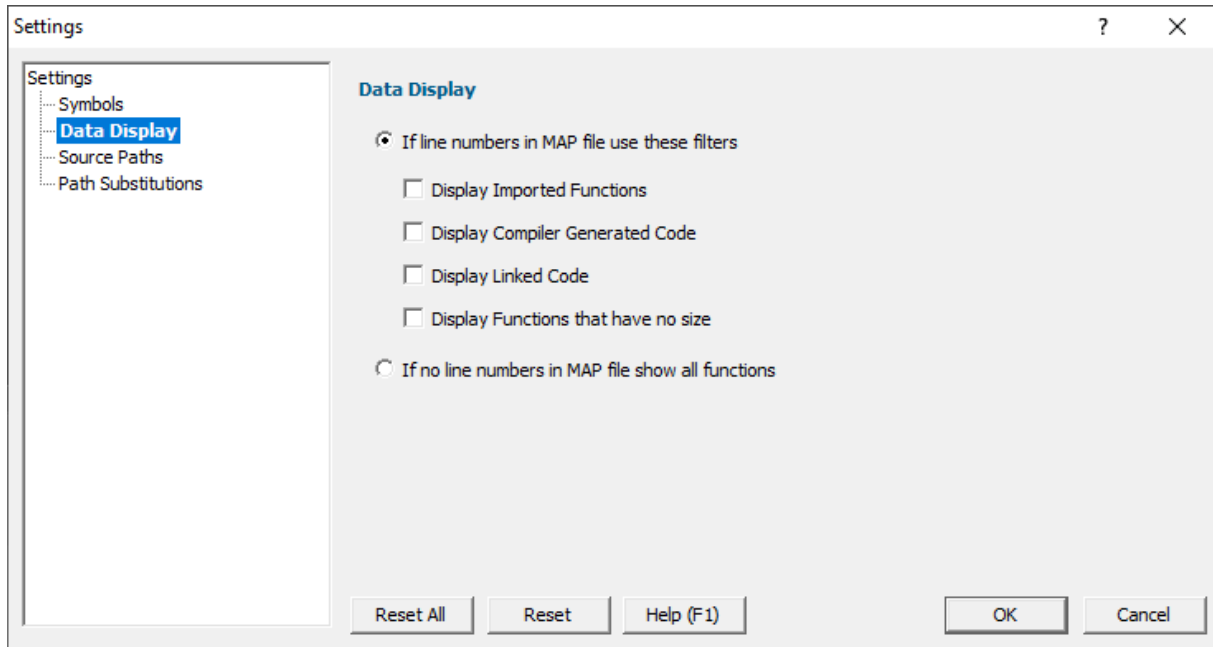
Depending upon the task you are performing you may wish to read the symbols as human readable symbols or compiler decorated C++ symbols.

**Reset** - Resets the settings on the current page.

**Reset All** - Resets **all** global settings, not just those on the current page.

## 7.2 Data display

The Symbols settings allow you to choose how symbols are displayed.



For MAP files that have line numbers you may wish to view additional information as well as the line numbers.

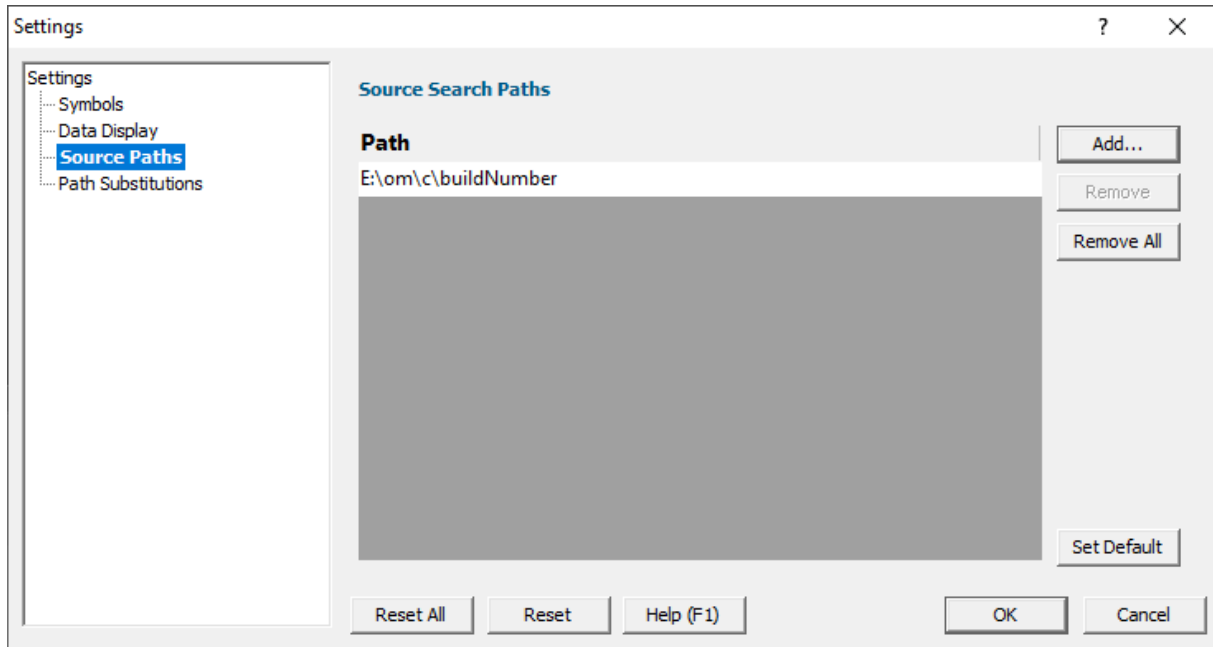
- **If line numbers in MAP file** > display all information relating to symbols that have line information
  - **Display Imported Functions** > functions imported from other DLLs will be displayed
  - **Display Compiler Generated Code** > compiler generated code will be displayed
  - **Display Linked Code** > linked object code from .obj and .lib will be displayed
  - **Display Functions that have no size** > symbols with no size will be displayed
- **If no line numbers in MAP file** > display all information that was found in the MAP file

**Reset** - Resets the settings on the current page.

**Reset All** - Resets **all** global settings, not just those on the current page.

## 7.3 Source code paths

The Source code paths settings allow you to specify where Map File Browsers looks for source code files.



## Manually adding path type directories

The Path list shows all the paths that will be searched for debug information in PDB files.

You can modify the list of files for each path type in the following ways:

- **Add** > appends a row to the directory list > enter the directory path

Edit a directory path by double clicking the entry. The usual controls apply for removing list items:

- **Remove** > removes selected items from the list
- **Remove All** > clears the list
- **Set Default** > populates the list with each directory found in the PATH environment variable

Alternatively, press **Del** to delete selected items, and **Ctrl** + **A** to select all items in the list first.

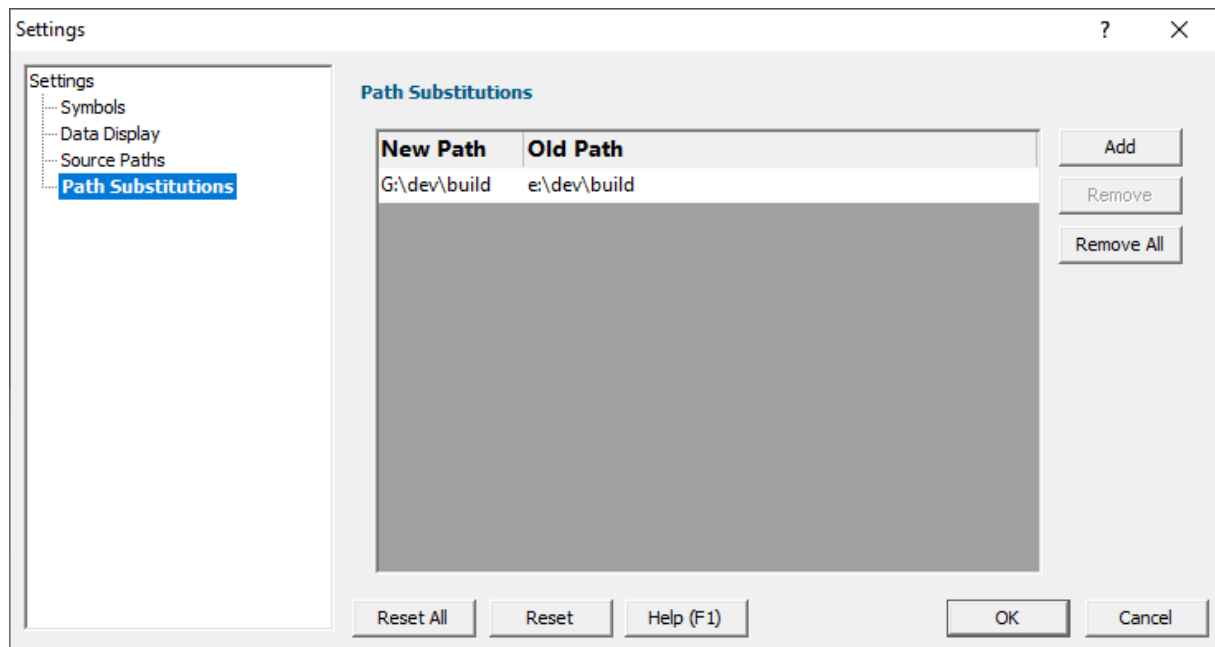
**Reset** - Resets the settings on the current page.

**Reset All** - Resets **all** global settings, not just those on the current page.

## 7.4 Path Substitutions

The **Path Substitutions** tab allows you to specify file path substitutions to handle copying builds from build machines to development or test machines .

The default settings are shown below:



### Path Substitutions

Some software development schemes have multiple rolling builds of their software, often enabled by using substituted disk drive naming schemes.

When you download the build to your development machine for development and testing, debugging information may reference disk drives that don't exist on your machine, for example, drive X: while your machine only has C:, D:, and E: drives.

Or you may just be copying a build from a drive on a development machine to a subdirectory on a drive on your test machine.

These options let you remap the substitution so that the MAP File Browser looks in the correct place for the source code.

- **Add** > adds a row to the **File Paths Substitutions** table > enter the new path that will replace the old path in the **New Path** column > click in the **Old Path** column > enter the path that is being replaced

For example, you might enter `c:\users\stephen\documents` for the new path and `f:\dev\build` for the old path.

You can double click to edit drives and paths in the table, or remove items:

- **Remove** > removes selected substitutions from the list
- **Remove All** > removes all substitutions from the list


Alternatively, press **Del** to delete selected items, and **Ctrl** + **A** to select all items in the list first.

#### Example: Changed disk drive

Project originally located at	m:\dev\build\testApp
Project copied to	e:\dev\build\testApp
New Path	e:\
Old Path	m:\

#### Example: Project copied to a new location

Project originally located at	f:\dev\build\testApp
Project copied to	C:\Users\Stephen\Documents\testApp
New Path	C:\Users\Stephen\Documents
Old Path	f:\dev\build

 The slashes do not have to match, a forward slash will match a backslash when comparing path fragments. This is deliberate - to improve ease of use with libraries built by different compilers (LLVM and compilers that use it use forward slashes, whereas Visual Studio etc use backslashes).

**Reset All** - Resets **all** global settings, not just those on the current page.

**Reset** - Resets the settings on the current page.

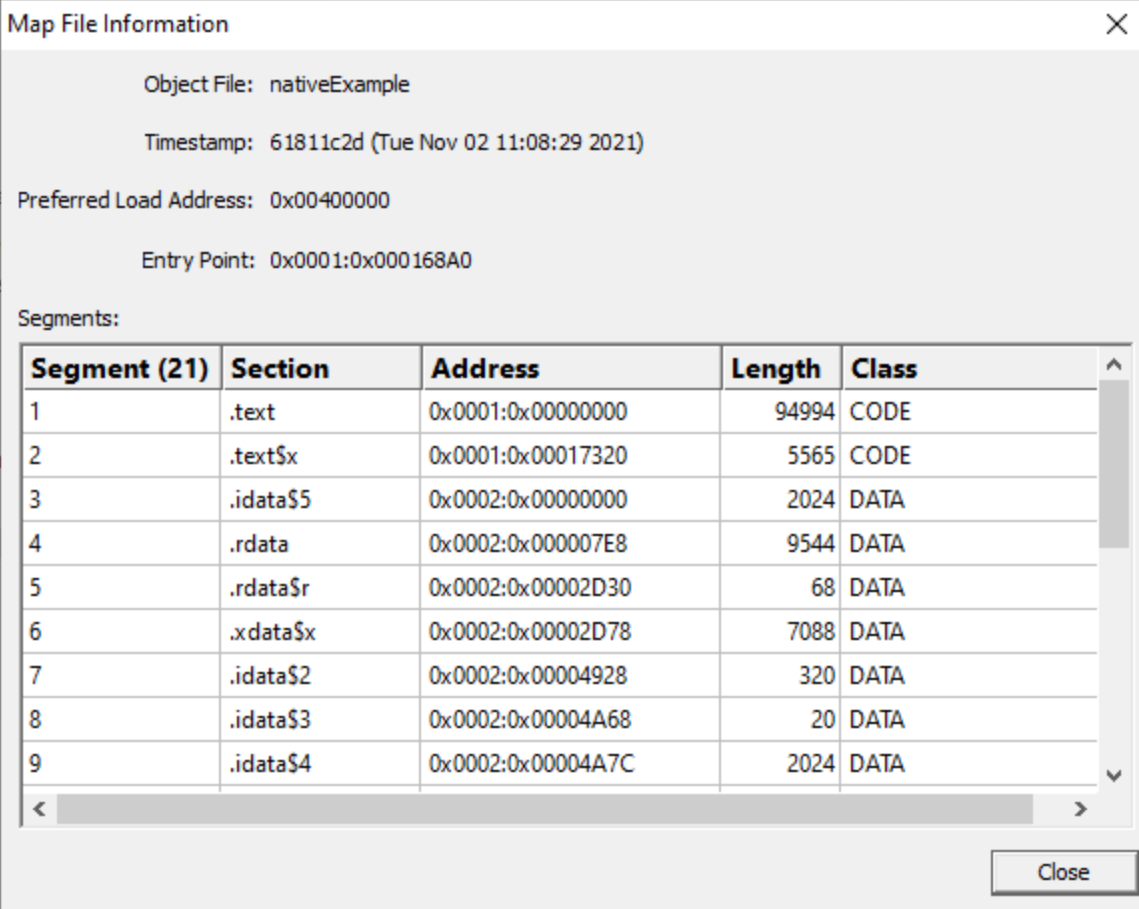
**Part**

**VIII**

## 8 MAP File Information

The MAP File Information dialog displays information about a MAP file that is not displayed by the main display.

This information includes the DLL/EXE name, timestamp, preferred load address, entry point and a list of the segments that make up the binary image.



Map File Information

Object File: nativeExample

Timestamp: 61811c2d (Tue Nov 02 11:08:29 2021)

Preferred Load Address: 0x00400000

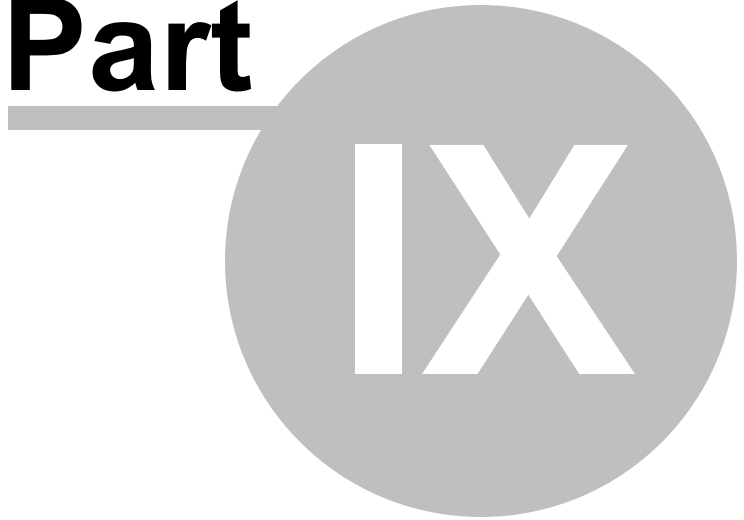
Entry Point: 0x0001:0x000168A0

Segments:

Segment (21)	Section	Address	Length	Class
1	.text	0x0001:0x00000000	94994	CODE
2	.text\$x	0x0001:0x00017320	5565	CODE
3	.idata\$5	0x0002:0x00000000	2024	DATA
4	.rdata	0x0002:0x000007E8	9544	DATA
5	.rdata\$r	0x0002:0x00002D30	68	DATA
6	.xdata\$x	0x0002:0x00002D78	7088	DATA
7	.idata\$2	0x0002:0x00004928	320	DATA
8	.idata\$3	0x0002:0x00004A68	20	DATA
9	.idata\$4	0x0002:0x00004A7C	2024	DATA

Close

**Part**



**IX**

## 9 How to use MapFileBrowser

### Load MAP File Information

Use the **File > Load MAP file...** option to load the appropriate MAP file.

The grid displays various attributes of each symbol in the MAP file. You can sort the grid by clicking the appropriate column header. Click the same header to reverse the sort order.

Select a symbol to see information about the line numbers and source code.

### Filtering

You can filter by name by typing the name into the **Name Filter** box and clicking the **Filter** button to perform the filtering.

### Viewing function data

As each item in the list is selected the Line Numbers are updated and the source code display updates to show the source code for the function. All lines in the function that contain executable code (as indicated by the line number information) are coloured grey. The current line for the function is coloured bright green.

### Querying data

You can query data by using the two Query fields below the main grid.

#### Relative query

Type the relative address (also known as address offset) into the Query by Offset field, then click Query. The symbol information is displayed.

The field accepts decimal or hexadecimal values. Hex values must be prefixed with 0x.

#### Absolute query

Type the absolute address into the Query by Address field, type the absolute DLL load address into the Alternate Load Address field, then click Query. The symbol information is displayed.

The fields accept decimal or hexadecimal values. Hex values must be prefixed with 0x.

## 9.1 Decoding an absolute crash address

### Scenario:

A customer has supplied you with a crash report containing a callstack with addresses. The callstack also indicates which module relates to which address.

The customer has also supplied you with a list of module load addresses.

### Example Data:

```
Exception code: C0000005 ACCESS_VIOLATION
Fault address: 0x005f5eec (base 0x00400000) C:\Program Files (x86)\Software Verification\
Exception Parameters:
    0: 0x00000000 [Read Error]
    1: 0x035f0034 [Address]
```

### Registers:

```
EAX:035F0034
EBX:00000000
ECX:FFFDD000
EDX:00002370
ESI:006F7D58
EDI:035F0034
CS:EIP:0023:005F5EEC
SS:ESP:002B:0018FE14 EBP:0018FE3C
DS:002B ES:002B FS:0053 GS:002B
Flags:00010202
```

### StackTrace

```
C:\Program Files (x86)\Software Verification\C++ Memory Validator\memoryValidator.exe : 0x035f0034
C:\Program Files (x86)\Software Verification\C++ Memory Validator\memoryValidator.exe : 0x00000000
C:\Windows\syswow64\msvcrt.dll : 0x75D70000 : 0x75D7C3E4
C:\Windows\syswow64\msvcrt.dll : 0x75D70000 : 0x75D836B6
C:\Program Files (x86)\Software Verification\C++ Memory Validator\memoryValidator.exe : 0x035f0034
C:\Windows\syswow64\kernel32.dll : 0x754D0000 : 0x754E3365
C:\Windows\SysWOW64\ntdll.dll : 0x77920000 : 0x77959F6D
C:\Windows\SysWOW64\ntdll.dll : 0x77920000 : 0x77959F40
C:\Windows\SysWOW64\ntdll.dll : 0x77920000 : 0x77959F40
```

This is data from a real crash a few years ago, from C++ Memory Validator 5.80.

### Question:

How do you decode these absolute addresses?

### Answer:

In the above data we can see a callstack containing entries for ntdll.dll, msvcrt.dll, and memoryValidator.exe.

All the modules are Microsoft DLLs except for the EXE, which is part of C++ Memory Validator, one of our tools.

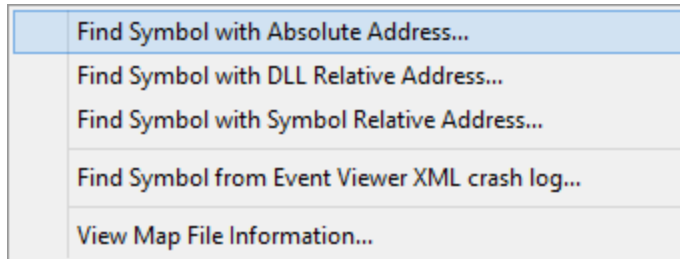
To decode these values, we load memoryValidator.map into MapFileBrowser, then for each symbol we take the following actions.

For our purposes here, we're going to show how to convert one symbol. We're going to use the first symbol from memoryValidator.exe in the example data above.

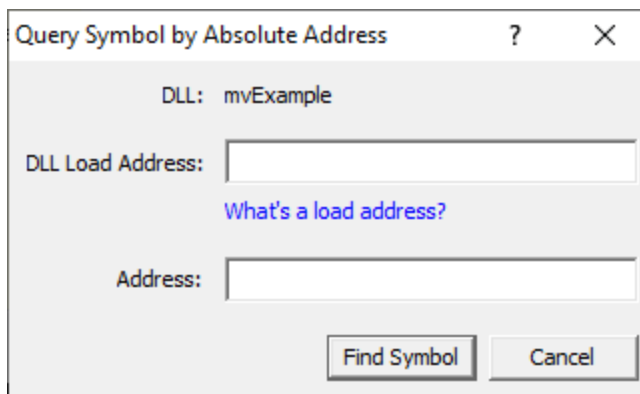
```
0x005f5eec (base 0x00400000)
```

The address is 0x005f5eec. The DLL loaded at 0x00400000. You'll notice the load address for all MemoryValidator.exe entries is 0x00400000.

From the Query menu choose **Find Symbol with Absolute Address...**



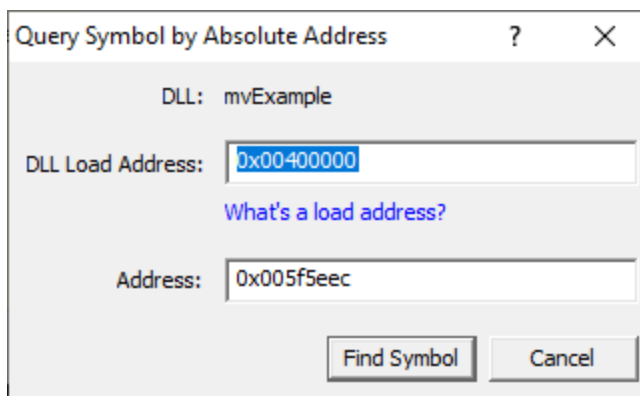
The Query Symbol by Absolute Address dialog is displayed.



Type the DLL load address into the DLL Load Address field. Prefix any hexadecimal addresses with 0x.

Type the symbol address into the Address field. Prefix any hexadecimal addresses with 0x.

Click the **Find Symbol** button.



The appropriate location in the code is found and displayed.

```

Filename: E:\om\c\memory32\abserv\MemorySettingData.cpp Line 5365
Function: private: int __thiscall MemorySettingData::saveSoftwareUpdateInformation(struct HKEY__ * const)
Address: 0x005F5EEC
Copy

5358 //---NAME-----
5359 //---DESCRIPTION-----
5360 //---PARAMETERS-----
5361 //---RETURN CODES-----
5362 //-----
5363
5364 int MemorySettingData::saveSoftwareUpdateInformation(const HKEY &hKey)
5365 {
5366     int ok = TRUE;
5367
5368     ok &= writeData(hKey, _T("suea"), softwareUpdateEmailAddress); // stephen@softwareverify.com
5369     ok &= writeData(hKey, _T("sup"), softwareUpdatePassword); // tester
5370     ok &= writeData(hKey, _T("sus"), (DWORD &)softwareUpdateSchedule);
5371     ok &= writeData(hKey, _T("sulc"), softwareUpdateLastCheck);
5372
5373     return ok;

```

**Results:**

Repeating the process for the data shown above resulted in this information.

```

0x005f5eec (base 0x00400000) C:\Program Files (x86)\Software Verification\C++ Memory Valid
C:\Program Files (x86)\Software Verification\C++ Memory Validator\memoryValidator.exe : 0x
C:\Program Files (x86)\Software Verification\C++ Memory Validator\memoryValidator.exe : 0x
C:\Windows\syswow64\msvcrt.dll : 0x75D70000 : 0x75D7C3E4
C:\Windows\syswow64\msvcrt.dll : 0x75D70000 : 0x75D836B6
C:\Program Files (x86)\Software Verification\C++ Memory Validator\memoryValidator.exe : 0x
C:\Windows\syswow64\kernel32.dll : 0x754D0000 : 0x754E3365
C:\Windows\SysWOW64\ntdll.dll : 0x77920000 : 0x77959F6D
C:\Windows\SysWOW64\ntdll.dll : 0x77920000 : 0x77959F40
C:\Windows\SysWOW64\ntdll.dll : 0x77920000 : 0x77959F40

```

**Help! I have a crash address but I don't know what the load address is? What do I do?**

You need to read about load addresses.

## 9.2 Decoding a relative crash address

**Scenario:**

A customer has supplied you with a crash report containing a callstack with relative offsets from DLLs. The callstack also indicates which module relates to which address.

**Example Data:**

```
Exception code: C0000005 ACCESS_VIOLATION
Fault offset: 0x00036FA3 C:\WINDOWS\system32\MSVCRT.dll
Exception Parameters:
  0: 0x00000000 [Read Error]
  1: 0x5f8f2000 [Address]
```

```
Registers:
EAX:B3BEB6D4
EBX:5F8CB6C8
ECX:150BE5B5
EDX:00000000
ESI:5F8F2000
EDI:01B98DEC
CS:EIP:001B:77C46FA3
SS:ESP:0023:0012F158 EBP:0012F160
DS:0023 ES:0023 FS:003B GS:0000
Flags:00010212
```

#### StackTrace

```
C:\WINDOWS\system32\MFC42u.DLL : 0x0000270a
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x000db989
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x000db1f8
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00121a83
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00121b7e
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00174ec5
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00175094
C:\WINDOWS\system32\MFC42u.DLL : 0x00013724
C:\WINDOWS\system32\MFC42u.DLL : 0x00014245
C:\WINDOWS\system32\MFC42u.DLL : 0x00001b31
C:\WINDOWS\system32\MFC42u.DLL : 0x0008cba7
```

This is data from a real crash many years ago.

#### Question:

There are no DLL load addresses and the addresses aren't addresses, but offsets from the start of a DLL. How do you decode these relative offsets?

#### Answer:

In the above data we can see a callstack containing entries for mfc42u.dll, and memoryValidator.exe.

All the modules are Microsoft DLLs except for the EXE, which is part of C++ Memory Validator, one of our tools.

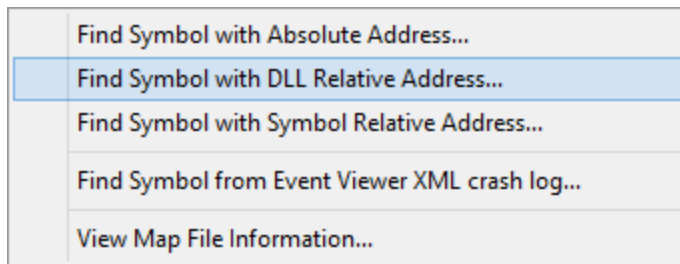
To decode these values, we load memoryValidator.exe into MapFileBrowser.exe, then for each symbol we take the following actions.

For our purposes here, we're going to show how to convert one symbol. We're going to use the first symbol from memoryValidator.exe in the example data above.

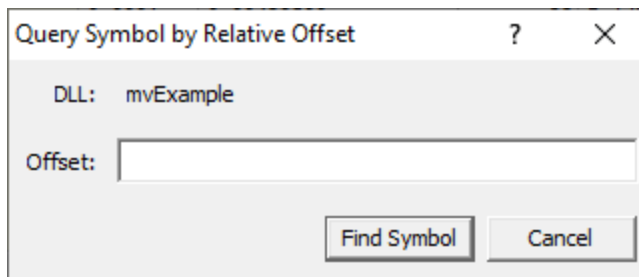
```
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x000db989
```

The relative address (or offset) is 0x000db989. We don't know the DLL load address.

From the Query menu choose **Find Symbol with DLL Relative Address...**

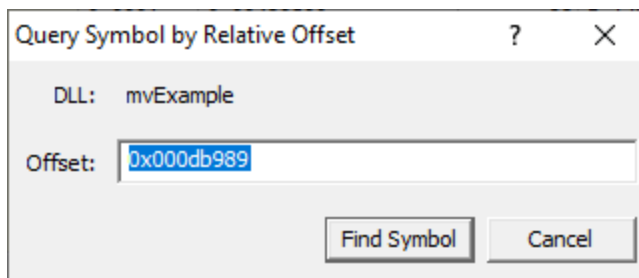


The Query Symbol by Absolute Address dialog is displayed.

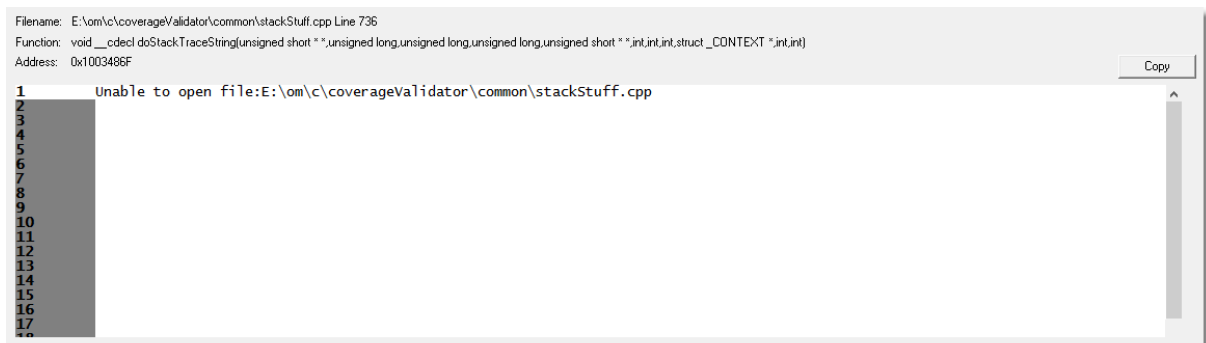


Type the relative address into the Offset field. Prefix any hexadecimal addresses with 0x.

Click the **Find Symbol** button.



The appropriate location in the code is found and displayed. In this example MapFileBrowser could not locate the source code (as the file location is not valid on this machine)



**Results:**

Repeating the process for the data shown above resulted in this information.

```
Exception code: C0000005 ACCESS_VIOLATION
Fault offset: 0x00036FA3 C:\WINDOWS\system32\MSVCRT.dll
Exception Parameters:
  0: 0x00000000 [Read Error]
  1: 0x5f8f2000 [Address]
```

```
Registers:
EAX:B3BEB6D4
EBX:5F8CB6C8
ECX:150BE5B5
EDX:00000000
ESI:5F8F2000
EDI:01B98DEC
CS:EIP:001B:77C46FA3
SS:ESP:0023:0012F158 EBP:0012F160
DS:0023 ES:0023 FS:003B GS:0000
Flags:00010212
```

StackTrace

```
C:\WINDOWS\system32\MFC42u.DLL : 0x0000270a
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x000db989
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x000db1f8
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00121a83
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00121b7e
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00174ec5
C:\Program Files\Software Verification\Memory Validator\memoryValidator.exe : 0x00175094
C:\WINDOWS\system32\MFC42u.DLL : 0x00013724
C:\WINDOWS\system32\MFC42u.DLL : 0x00014245
C:\WINDOWS\system32\MFC42u.DLL : 0x00001b31
C:\WINDOWS\system32\MFC42u.DLL : 0x0008cba7
```

## 9.3 Decoding a symbol relative crash address

### Scenario:

A customer has supplied you with a crash report containing a callstack with symbol relative offsets from DLLs. The callstack also indicates which module relates to which address.

### Example Data:

```

ntoskrnl.exe!KeSynchronizeExecution+0x2246
ntoskrnl.exe!KeWaitForMultipleObjects+0x135e
ntoskrnl.exe!KeWaitForMultipleObjects+0xdd9
ntoskrnl.exe!KeWaitForSingleObject+0x373
ntoskrnl.exe!KeStallWhileFrozen+0x1977
ntoskrnl.exe!_misaligned_access+0x13f9
ntoskrnl.exe!KeWaitForMultipleObjects+0x152f
ntoskrnl.exe!KeWaitForMultipleObjects+0xdd9
ntoskrnl.exe!KeWaitForSingleObject+0x373
ntoskrnl.exe!NtWaitForSingleObject+0xb2
ntoskrnl.exe!setjmpex+0x34a3
ntdll.dll!ZwWaitForSingleObject+0xa
KERNELBASE.dll!WaitForSingleObjectEx+0x98
svlcoveragevalidatorstub.dll!sendCommandLineAndStartTimeToGUI+0x2868
svlcoveragevalidatorstub.dll!setValidatorFeedbackHookingComplete+0x1fa6
svlcoveragevalidatorstub.dll!svl_sendMessageRawToUserInterface+0x21837
svlcoveragevalidatorstub.dll!svl_sendMessageRawToUserInterface+0x218cb
KERNEL32.DLL!BaseThreadInitThunk+0x22
ntdll.dll!RtlUserThreadStart+0x34

```

This is real data from a bug at Software Verify Ltd. This is one thread from many in a dump relating to a deadlock bug we were investigating.

**Question:**

How do you decode these symbol relative offsets?

**Answer:**

In the above data we can see a callstack containing entries for ntoskrnl.exe, ntdll.dll, kernelbase.dll, kernel32.dll and svlcoveragevalidatorstub.dll.

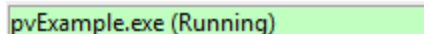
All the modules are Microsoft DLLs except for one DLL, which is part of C++ Coverage Validator, one of our tools.

To decode these values, we load svlCoverageValidatorStub.dll into MapFileBrowser, then for each symbol we take the following actions.

For our purposes here, we're going to show how to convert one symbol. We're going to use the first symbol from svlCoverageValidatorStub.dll in the example data above.

```
svlcoveragevalidatorstub.dll!sendCommandLineAndStartTimeToGUI+0x2868
```

Type the symbol name into the **Name Filter** field, then click **Filter**. This makes it easy to find the symbol we want.

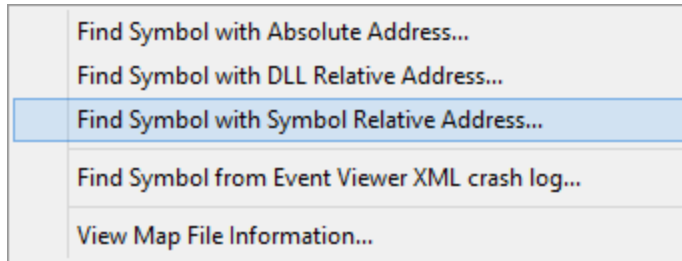


Once we have found the symbol, right click on the symbol to display the context menu and choose **Offset from this symbol...**

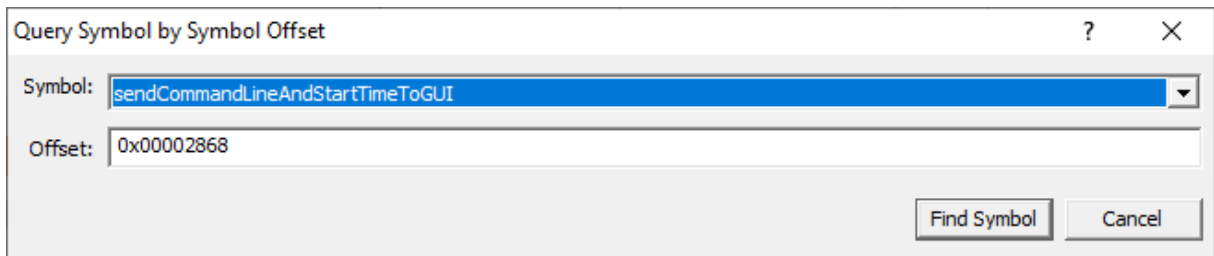


An alternate method is to click on the symbol to select it, then from the Query menu choose **Find Symbol with Symbol Relative Address...**

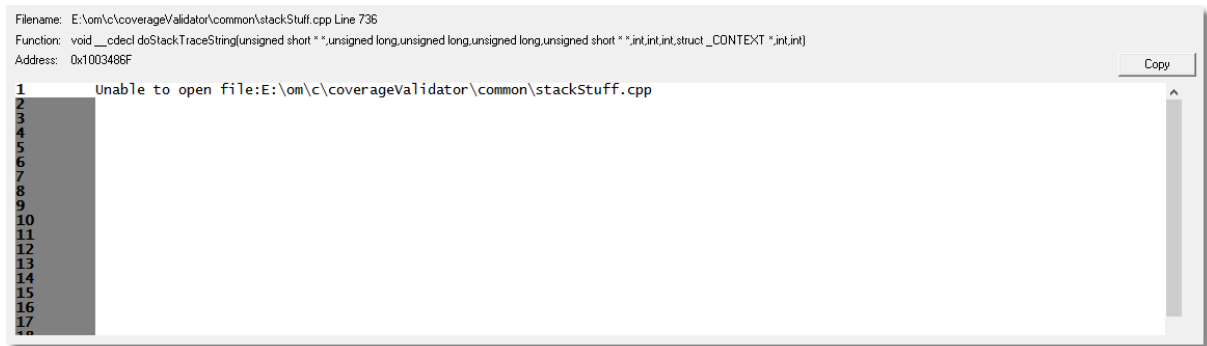
Or, from the Query menu choose **Find Symbol with Symbol Relative Address...** then choose the symbol you want from the combo box.



Type the offset into the dialog (hex values must be prefixed with 0x) and click OK.



The appropriate location in the code is found and displayed.



**Results:**

Repeating the process for the data shown above resulted in this information.

```
svlcoveragevalidatorstub.dll!sendCommandLineAndStartTimeToGUI+0x2868
svlcoveragevalidatorstub.dll!setValidatorFeedbackHookingComplete+0x1fa6
svlcoveragevalidatorstub.dll!svl_sendMessageRawToUserInterface+0x21837
svlcoveragevalidatorstub.dll!svl_sendMessageRawToUserInterface+0x218cb
```

doStackT  
stubSend  
memcpy  
wcsncpy

## 9.4 Decoding an Event Viewer XML crash log

**Scenario:**

A customer has supplied you with data from Windows Event Viewer about a crash. The log contains XML and you don't know which values are relevant.

The event log data will have a provider name of "Windows Error Reporting" or "Application Error".

The XML data is found on the "Details" tab with the XML View radio box selected.

#### Example Data:

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Application Error" />
    <EventID Qualifiers="0">1000</EventID>
    <Level>2</Level>
    <Task>100</Task>
    <Keywords>0x8000000000000000</Keywords>
    <TimeCreated SystemTime="2020-02-11T10:42:39.000000000Z" />
    <EventRecordID>260330</EventRecordID>
    <Channel>Application</Channel>
    <Computer>hydra</Computer>
    <Security />
  </System>
  <EventData>
    <Data>testDeliberateCrashVS6.exe</Data>
    <Data>1.0.0.1</Data>
    <Data>5e42850d</Data>
    <Data>testDeliberateCrashVS6.exe</Data>
    <Data>1.0.0.1</Data>
    <Data>5e42850d</Data>
    <Data>c0000005</Data>
    <Data>00001d07</Data>
    <Data>1490</Data>
    <Data>01d5e0c7fa70e745</Data>
    <Data>E:\om\c\testApps\testDeliberateCrashVS6\Debug\testDeliberateCrashVS6.exe</Data>
    <Data>E:\om\c\testApps\testDeliberateCrashVS6\Debug\testDeliberateCrashVS6.exe</Data>
    <Data>390bde30-4cbb-11ea-83d3-001e4fdb3956</Data>
    <Data />
    <Data />
  </EventData>
</Event>
```

This is data from a test program that is designed to crash.

#### Question:

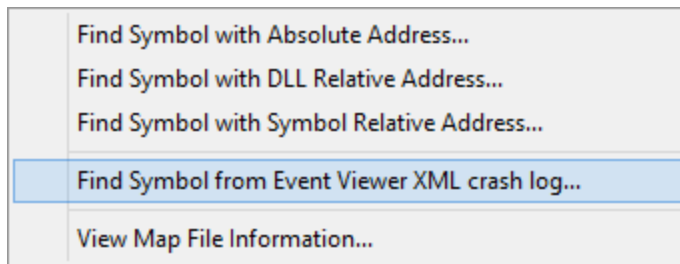
There the event log indicates a DLL, but no load address, two different addresses, an exception code and an offset from the start of the DLL. How do you decode this relative offset?

#### Answer:

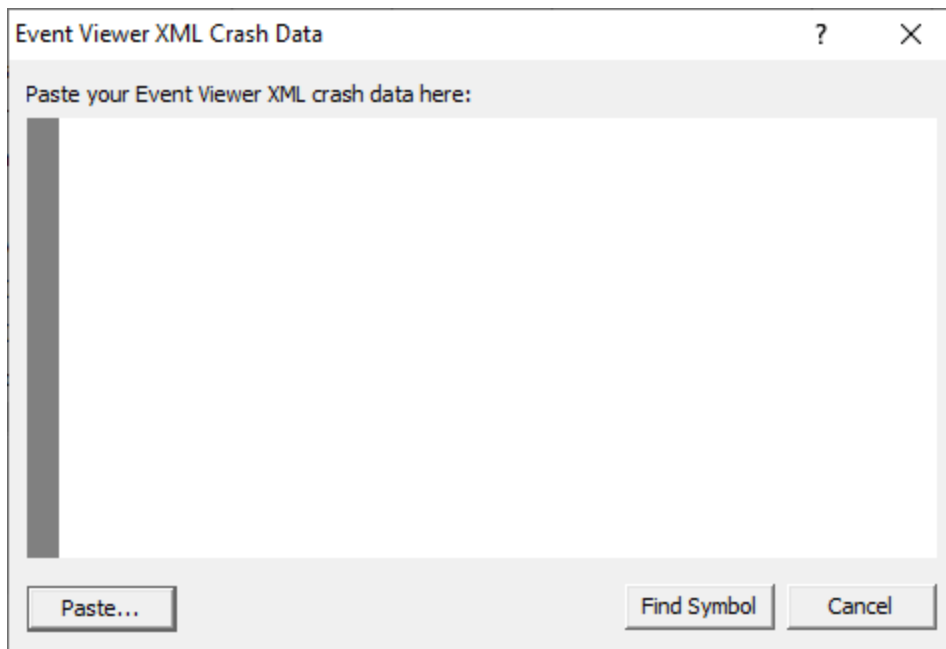
MapFileBrowser has an option specifically for this occasion.

The XML data indicates the crash happened in **testDeliberateCrashVS6.exe**. Load this into MapFileBrowser being sure to load the correct build version and that the PDB file can be found so that symbols get loaded.

From the Query menu choose **Find Symbol from Event Viewer XML crash log...**

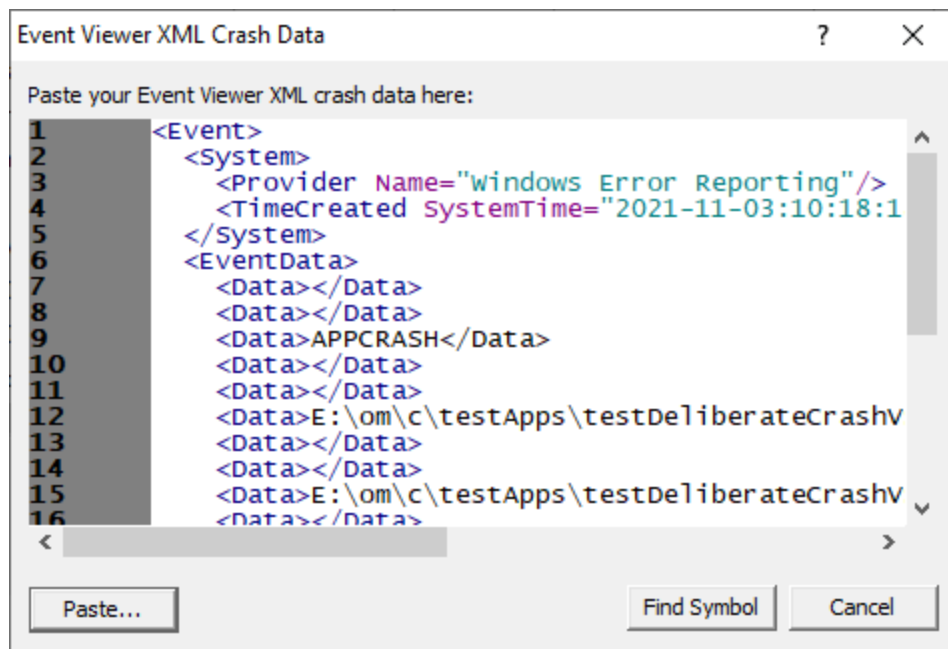


The Query Symbol by Absolute Address dialog is displayed.



Paste the XML data from the Event Viewer into the text field.

Click the **Find Symbol** button.



The appropriate location in the code is found and displayed.

```
Filename: E:\om\c\testApps\testDeliberateCrashVS6\testDeliberateCrashVS6Dlg.cpp Line 178
Function: private: void __thiscall CTestDeliberateCrashVS6Dlg::causeACrash(void)
Address: 0x004029C0
```

```
170     }
171
172     void CTestDeliberateCrashVS6Dlg::OnOK()
173     {
174         causeACrash();
175     }
176
177     void CTestDeliberateCrashVS6Dlg::causeACrash()
178     {
179         char *ptr = NULL;
180
181         *ptr = '\0';    // crash
182     }
183
184
185
186
```

## 9.5 What is a load address?

A load address is the address at which a DLL loads.

All versions of Microsoft Windows load modules (.dll, .exe) into address space that is reserved using a call to VirtualAlloc().

The allocation of VirtualAlloc() can be queried by calling Win32 API GetSystemInfo() and examining the value returned in **dwAllocationGranularity**. For all versions of Microsoft Windows this has been 64KB.

### **Why is the load address important?**

The load address is important because without it we can't calculate the offset inside the DLL so that we can obtain a symbol.

That's why a crash address with no DLL Load Address isn't very useful - we don't know which DLL the crash is in, nor do we know where the DLL was loaded.

### **But I don't have a load address. What can I do?**

Depending upon how your module (DLL/EXE) was built we may be able to guess the correct load address.

If the OS you are using is Windows XP or earlier, we can guess the address.

### **First a brief chat about Address Space Layout Randomisation...**

If the OS you are using is Windows Vista or later, we may be able to guess the load address. The reason this is not precise is because something known as Address Space Layout Randomisation (ASLR) was introduced with Microsoft Vista to improve security against many malicious computer attacks. Any program built with ASLR enabled when run on Vista (or later) will have the load address for all modules (including the .exe) randomised, making guessing the load address a waste of time.

ASLR is enabled by the /DYNAMICBASE in the linker settings of Visual Studio.

If you are using Visual Studio 2005 or earlier this setting is not available, your program is not affected by ASLR.

If you are using Visual Studio 2008 or later you will need to check to see if this option is present. If it is not present, your program is not affected by ASLR.

If you are not using Visual Studio to build your program then you may not be affected by this option, consult your compiler/linker documentation.

### **If your program is not affected by ASLR...**

We can try to guess the load address of your DLL/EXE. We can do this regardless of which compiler/linker you used to build your program. All the programs I mention here are free to download at the time of writing this help file.

### **VM Validator**

<https://www.softwareverify.com/cpp-virtual-memory.php>

This works for 32 bit and 64 bit programs.

## Method 1

- Start your program using VM Validator or attach to your running program with VM Validator.
- On the Summary tab, inspect the DLLs sub tab in the lower half of the display.
- Find the DLL name in the DLL column.
- The load address is the value in the Address column.

DLLs	Page Faults						
DLL (133)	Fault Count	Address	Size	Commit	Reserve	CPU	
E:\om\c\dbgHelpBrowser\Release\x86\dbgHelpBrowser.exe	0	0x00400000	1176.00 KB	1176.00 KB	0.00 KB	x86	
E:\om\c\testApps\testDeliberateCrash\Release\testDeliberateCrash.exe	0	0x00640000	100.00 KB	100.00 KB	0.00 KB	x86	
E:\om\c\dbgHelpBrowser\Release\x86\svlPEInfo.dll	0	0x006C0000	172.00 KB	172.00 KB	0.00 KB	x86	

## Method 2

- Start your program using VM Validator or attach to your running program with VM Validator.
- Go to the Paragraphs tab.
- Find any purple entry, check the DLL name in the Description field.
- The load address is the value in the Address column.

Summary		Virtual		Pages		Paragraphs	
Address	Size	Type	Protect	Working Set	Shared	Swap	Description
0x002C0000	64 KB	Private	Read, Write	Read/write.			Committed, Reserved
0x002D0000	152 KB	Private					Reserved, Committed, Reserved
0x00300000	1,024 KB	Private					Reserved
0x00400000	1,176 KB	Image	Read Only	Read-only, Executable and read-only.	Shared: 98		e:\om\c\dbghelpbrowser\release\x86\dbghelpbrowser.exe
0x00530000	796 KB	Mapped	Read Only	Read-only.	Shared: 49		Committed, Free

In the example above, for dbgHelpBrowser.exe, the load address is 0x00400000.

## Process Explorer

<https://technet.microsoft.com/en-us/sysinternals/processexplorer.aspx>

This works for 32 bit and 64 bit programs.

- Start your program
- Start Process Explorer. *If your program is a service or runs as administrator you'll need to start Process Explorer as administrator.*
- In Process Explorer, enable View -> Show Lower Pane. Then for View -> Lower Pane Window, choose DLLs.
- Select your program in the top window.
- Find your DLL in the bottom window. Right click. Choose Properties from the Context menu.

Process Explorer - Sysinternals: www.sysinternals.com [ZEUS\Stephen]

File Options View Process Find DLL Users Help

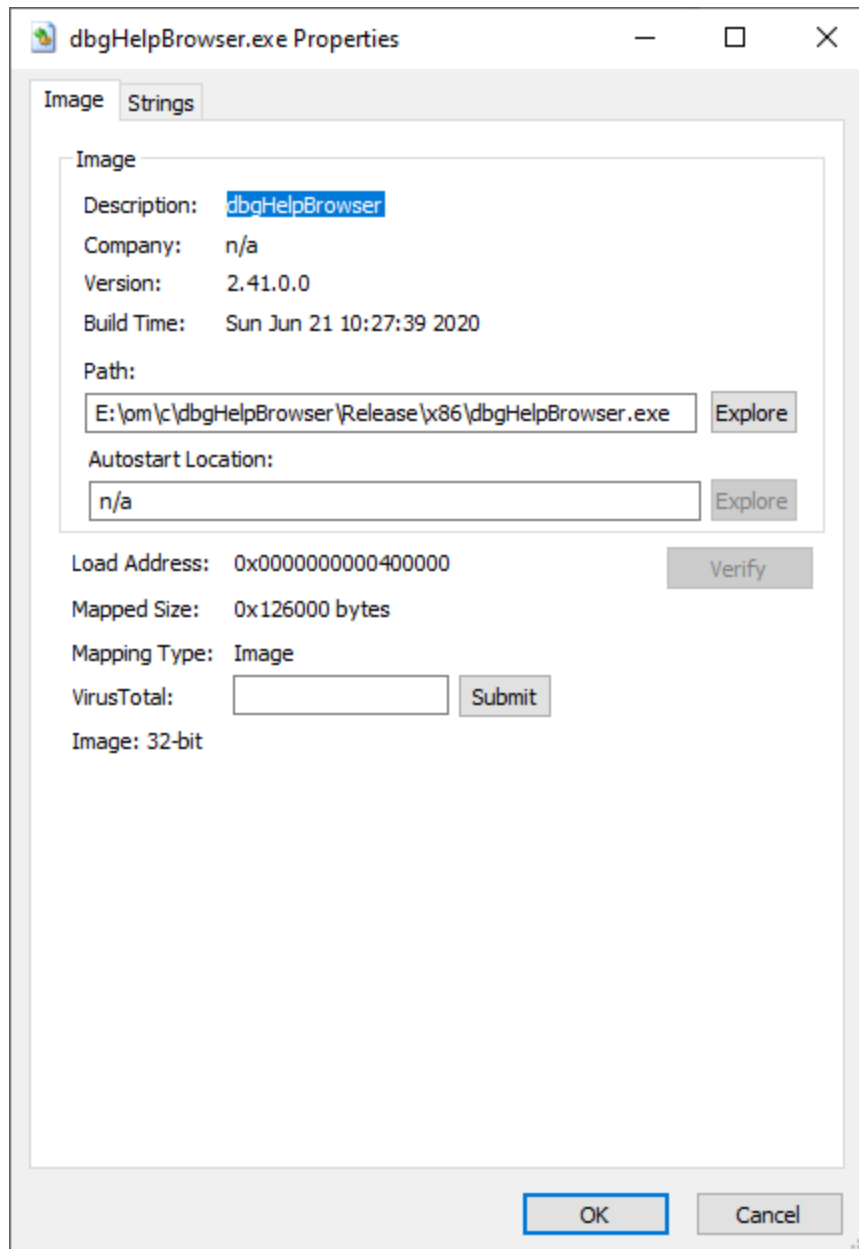
Process	CPU	Private Bytes	Working Set	PID	Description	Company Name
SecurityHealthSystray.exe		1,712 K	8,704 K	11880	Windows Security notificatio...	Microsoft Corporation
OneDrive.exe		29,012 K	70,728 K	12044	Microsoft OneDrive	Microsoft Corporation
WZQKPICK.EXE		1,496 K	8,108 K	6852	WinZip Executable	WinZip Computing, Inc.
notepad.exe		3,112 K	16,952 K	308	Notepad	Microsoft Corporation
svllauncher.exe		4,696 K	12,588 K	1052	svlLauncher	Software Verify Limited - w...
dbgHelpBrowser.exe	12.47	28,420 K	51,624 K	6840	dbgHelpBrowser	Software Verify Limited - w...
eventlogcrashbrowser.exe	12.48	5,896 K	16,708 K	8836	Event Log Crash Browser	Software Verify Limited - w...
vmValidator.exe	6.14	9,664 K	21,964 K	6188	vmValidator	Software Verify Limited - w...
HELPMAN.EXE	0.23	108,860 K	186,692 K	5396	Help & Manual	EC Software GmbH

Name	Description	Company Name	Path
DataExchange.dll	Data exchange	Microsoft Corporation	C:\Windows\SysWOW64\DataExchange.dll
davclnt.dll	Web DAV Client DLL	Microsoft Corporation	C:\Windows\SysWOW64\davclnt.dll
davhlpr.dll	DAV Helper DLL	Microsoft Corporation	C:\Windows\SysWOW64\davhlpr.dll
dbgHelp.dll	Windows Image Helper	Microsoft Corporation	E:\om\c\dbgHelpBrowser\Release\x86\dbgHelp.dll
dbgHelpBrowser.exe	dbgHelpBrowser	Microsoft Corporation	E:\om\c\dbgHelpBrowser\Release\x86\dbgHelpBrowser.exe
dcomp.dll	Microsoft DirectComposition Library	Microsoft Corporation	C:\Windows\SysWOW64\dcomp.dll
DevDisptemProvider.dll	DeviceItem inproc devquery subsy...	Microsoft Corporation	C:\Windows\SysWOW64\DevDisptemProvider.dll
devobj.dll	Device Information Set DLL	Microsoft Corporation	C:\Windows\SysWOW64\devobj.dll
dlnashext.dll	DLNA Namespace DLL	Microsoft Corporation	C:\Windows\SysWOW64\dlnashext.dll
dprov.dll	Microsoft Remote Desktop Sessio...	Microsoft Corporation	C:\Windows\SysWOW64\dprov.dll

CPU Usage: 33.45% Commit Charge: 28.15% Processes: 198

- In the Properties dialog, read the load address.

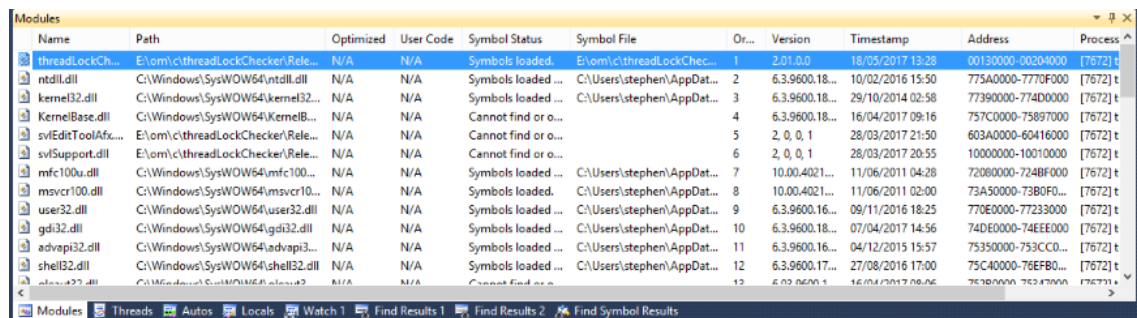


In the example above, for dbgHelpBrowser.exe, the load address is 0x00400000.

**Visual Studio** (any version)  
<https://www.visualstudio.com/>

- Start Visual Studio.
- From the Project menu, choose File -> Open -> Solution. Choose your executable.
- From the Debug menu, choose Start Debugging.
- From the Debug menu, choose Windows -> Modules.

- In the Modules window, find your DLL, then read the Address column.



Name	Path	Optimized	User Code	Symbol Status	Symbol File	Or...	Version	Timestamp	Address	Process
threadLockCh...	E:\om\c\threadLockChecker\Rele...	N/A	N/A	Symbols loaded...	E:\om\c\threadLockChec...	1	2.01.0.0	18/05/2017 13:28	00130000-00204000	[7672] t
ntdll.dll	C:\Windows\SysWOW64\ntdll.dll	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	2	6.3.9600.18...	10/02/2016 15:50	775A0000-7770F000	[7672] t
kernel32.dll	C:\Windows\SysWOW64\kernel32...	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	3	6.3.9600.18...	29/10/2014 02:58	77390000-774D0000	[7672] t
KernelBase.dll	C:\Windows\SysWOW64\KernelB...	N/A	N/A	Cannot find or o...		4	6.3.9600.18...	16/04/2017 09:16	757C0000-75897000	[7672] t
svlEditToolAfx...	E:\om\c\threadLockChecker\Rele...	N/A	N/A	Cannot find or o...		5	2, 0, 0, 1	28/03/2017 21:50	603A0000-60416000	[7672] t
svlSupport.dll	E:\om\c\threadLockChecker\Rele...	N/A	N/A	Cannot find or o...		6	2, 0, 0, 1	28/03/2017 20:55	10000000-10010000	[7672] t
mfc100u.dll	C:\Windows\SysWOW64\mfc100...	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	7	10.00.4021...	11/06/2011 04:28	72080000-724BF000	[7672] t
msvcr100.dll	C:\Windows\SysWOW64\msvcr10...	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	8	10.00.4021...	11/06/2011 02:00	73A50000-73B0F0...	[7672] t
user32.dll	C:\Windows\SysWOW64\user32.dll	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	9	6.3.9600.16...	09/11/2016 18:25	770E0000-77233000	[7672] t
gdi32.dll	C:\Windows\SysWOW64\gdi32.dll	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	10	6.3.9600.18...	07/04/2017 14:56	74DE0000-74EEE000	[7672] t
advapi32.dll	C:\Windows\SysWOW64\advapi3...	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	11	6.3.9600.16...	04/12/2015 15:57	75350000-753CC0...	[7672] t
shell32.dll	C:\Windows\SysWOW64\shell32.dll	N/A	N/A	Symbols loaded...	C:\Users\stephen\AppData...	12	6.3.9600.17...	27/08/2016 17:00	75C40000-76EFB0...	[7672] t
ole32.dll	C:\Windows\SysWOW64\ole32...	N/A	N/A	Cannot find or o...		13	6.3.9600.17...	16/04/2017 08:06	76380000-763A7000	[7672] t

In the example above, for threadLockChecker.exe, the load address is 0x00130000.

## WinDbg

[https://msdn.microsoft.com/en-gb/library/windows/hardware/ff551063\(v=vs.85\).aspx](https://msdn.microsoft.com/en-gb/library/windows/hardware/ff551063(v=vs.85).aspx)

- Start WinDbg
- From the File menu, choose Open Executable. Choose your executable.
- Type lm, then press return.
- All modules are listed. Find your module. The start address is the load address.

```

E:\om\c\dbgHelpBrowser\Release\x86\dbgHelpBrowser.exe - WinDbg:10.0.19041.1 X86
File Edit View Debug Window Help
Command
eip=770aeaa2 esp=0019fa20 ebp=0019fa4c iopl=0         nv up ei pl zr na pe nc
cs=0023  ss=002b  ds=002b  es=002b  fs=0053  gs=002b             efl=00000246
ntdll!LdrpDoDebuggerBreak+0x2b:
770aeaa2 cc          int     3
0:000> lm
start      end          module_name
001d0000 001fb000    svlPeInfo   (deferred)
00400000 00526000    dbgHelpBrowser (deferred)
10000000 1000f000    svlSupport  (deferred)
73400000 73569000    gdiplus     (deferred)
73650000 736b9000    MSVCP100    (deferred)
737a0000 7382d000    COMCTL32    (deferred)
73830000 73836000    MSIMG32     (deferred)
73ee0000 7431f000    mfc100u     (deferred)
74350000 7440f000    MSVCR100    (deferred)
74420000 74428000    VERSION     (deferred)
747c0000 747ca000    CRYPTBASE   (deferred)
747d0000 747f0000    SspiCli     (deferred)
747f0000 74866000    sechost     (deferred)
74870000 74967000    ole32       (deferred)
74ae0000 74b9b000    RPCRT4      (deferred)
74ba0000 74baf000    kernel appcore (deferred)
74bb0000 74c29000    ADVAPI32    (deferred)
74c40000 74c9f000    bcryptPrimitives (deferred)
74d00000 74d17000    win32u      (deferred)
74d20000 752e6000    windows storage (deferred)
752f0000 752f6000    PSAPI       (deferred)
754e0000 75524000    SHLWAPI     (deferred)
75640000 757d7000    USER32     (deferred)
757e0000 758c0000    KERNEL32    (deferred)
75dd0000 75e8f000    msvcrt      (deferred)
75e90000 76409000    SHELL32     (deferred)
76480000 76504000    shcore      (deferred)
76510000 76531000    GDI32       (deferred)
76540000 7655b000    profapi     (deferred)
76570000 766cb000    qdi32full   (deferred)
766d0000 767ef000    ucrtbase    (deferred)
767f0000 767fd000    UMPDC       (deferred)
76800000 76813000    cryptsp     (deferred)
76930000 769ac000    msvcp win   (deferred)
769e0000 76c55000    combase     (deferred)
76c60000 76cf2000    OLEAUT32    (deferred)
76d00000 76d43000    powrprof    (deferred)
76d50000 76d8b000    cfqmgr32    (deferred)
76d90000 76f8e000    KERNELBASE   (deferred)
77000000 7719a000    ntdll       (pdb symbols)  C:\ProgramData\dbg\sym\wntdl
7b330000 7b451000    dbghelp     (deferred)
7ba00000 7ba7a000    svlEditToolAfx (deferred)
0:000>
Ln 0, Col 0 Sys 0:<Local> Proc 000:1bf0 Thrd 000:18dc ASM OVR CAPS NUM

```

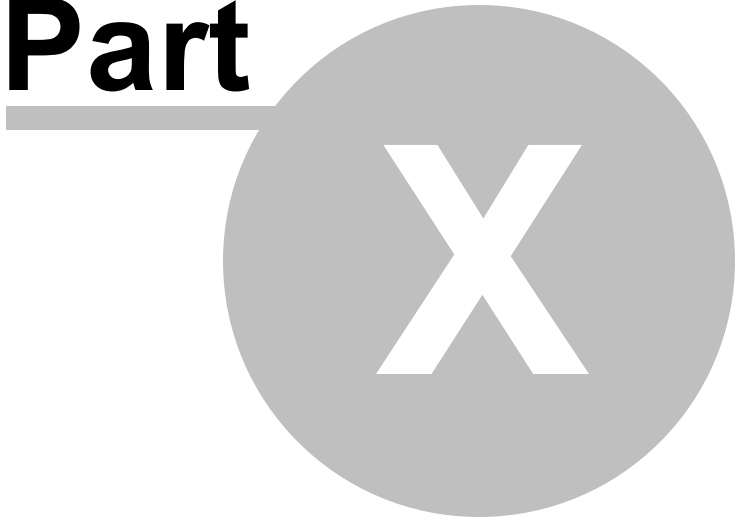
In the example above, for threadLockChecker.exe, the load address is 0x00130000.

### Final Comments

OK, you should now know how to find the load address of a DLL or an EXE (or any module type). Remember that a load address obtained this way is only valid for symbol decoding if the executable doesn't have ASLR applied to it.

If your crash reporting code only grabs crash addresses and not DLL load addresses, you need to update your code so that you grab DLL load addresses at the time of the crash. That way you know for sure what the load addresses were and you won't have to guess the load addresses in future.

**Part**



## 10 Command Line Interface

MapFileBrowser can be used from the command line as well as with the GUI.

The command line options allow you to view debug information that is in a MAP file, and optionally highlight a symbol at a specified offset.

### **/fileName**

Specifies the module to load. This is typically a .exe or a .dll.

`/fileName path-to-executable`

Example: `/fileName e:\om\c\test\release\test.exe`

### **/offset**

Specifies an offset inside the executable. MapFileBrowser will highlight the symbol that occupies this location.

Typically this offset will be calculated from a crash location.

For example:

If a DLL is loaded at 0x00400000 and a crash happens at 0x00420192, the offset is calculated by subtracting the DLL load address from the crash address.

That is:  $0x00420192 - 0x00400000$ , which gives 0x00020192.

The offset is 0x00020192.

The offset must be specified in hexadecimal with a leading 0x.

`/offset value`

Example: `/offset 0x00020192`

### **Example Command Line**

#### **32 bit applications**

```
MapFileBrowser.exe /fileName e:\test\release\test.map /offset 0x00020192
```

#### **64 bit applications**

```
MapFileBrowser_x64.exe /fileName e:\test\release\test.map /offset 0x00020192
```



