Exception Tracer

by

Software Verify

Software Verify

Troubleshoot your software

Exception Tracer

View all exceptions thrown by an application

by Software Verify Limited

Welcome to the Exception Tracer software tool.

Exception Tracer is a software tool that allows you to monitor all exceptions thrown by an application as it executes.

You can optionally automatically single step the application through the code, and create minidumps when exceptions are thrown, and when the application exits.

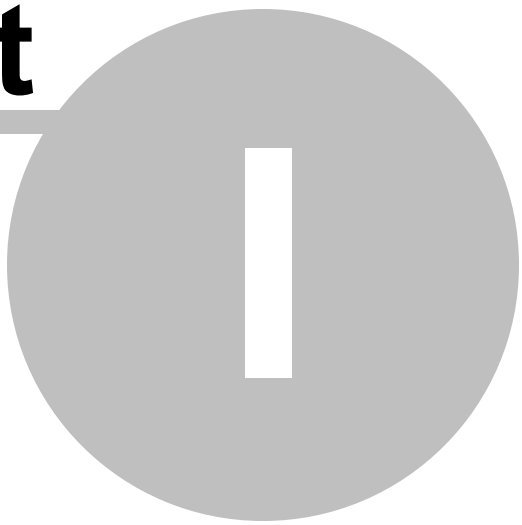
We hope you will find this document useful.

Exception Tracer Help

Printed: December 2023 in United Kingdom.

Table of Contents

Part



1 How to get Exception Tracer

Exception Tracer is free for commercial use. Exception Tracer can be downloaded for Software Verify's website at https://www.softwareverify.com/product/exception-tracer/.

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

Windows Help	https://www.softwareverify.com/documentation/chm/exceptionTracer.chm
PDF	https://www.softwareverify.com/documentation/pdfs/exceptionTracer.pdf
Online	https://www.softwareverify.com/documentation/html/exceptionTracer/index.html

Whilst Exception Tracer is free for commercial use, Exception Tracer 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.

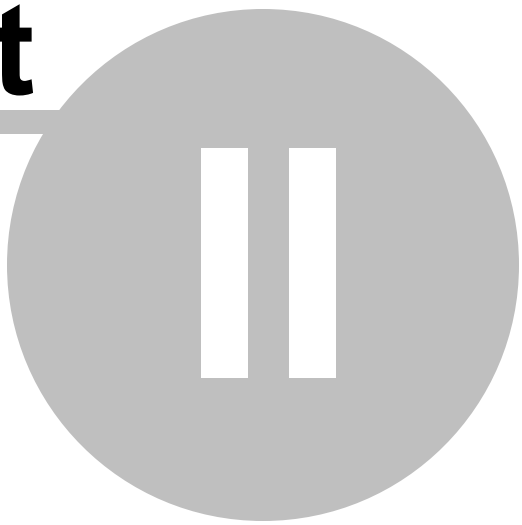
Contact Software Verify at:

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

email	sales@softwareverify.com
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 Exception Tracer do?

Exception Tracer allows you to monitor all exceptions thrown by an application.

This can be very useful for monitoring and debugging exceptions that are thrown by an application that would be quite cumbersome and clumsy with a regular debugger. A regular debugger would insist on breaking into the debugger because an exception was thrown, which is useful for many, if not most scenarios. But sometimes you just want to watch what is happening so that you can view application behaviour in the whole and then drill down into the events that have been captured, and that is what Exception Tracer is for.

Exception Tracer includes an option to allow you to automatically single step the application, with varying levels of data collection, so that you can debug things like stack overflows which are often impossible to debug with a traditional debugger (which just shows you endless frames of recursing calls but can't show you the root of the callstack because it's not designed to show you callstacks that deep).

Exception Tracer can also create minidumps when exceptions occur and at process exit.

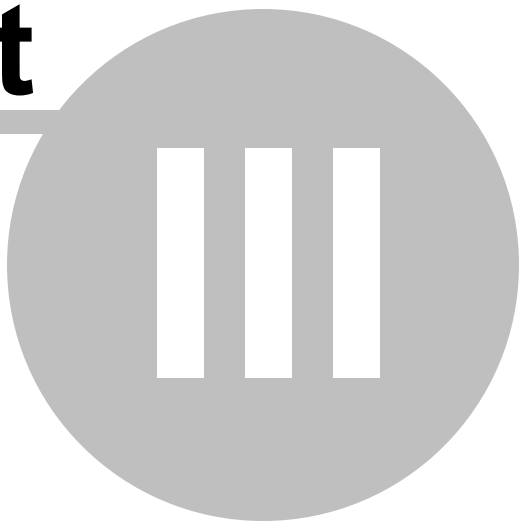
32 bit and 64 bit

32 bit and 64 bit applications are supported. On 64 bit Operating systems if a 64 bit executable is started by the 32 bit Exception Tracer, the 64 bit Exception Tracer is automatically started.

History

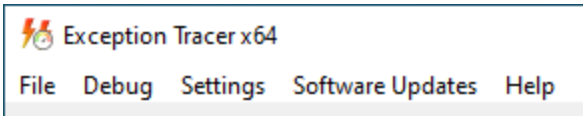
Exception Tracer has been an internal tool at Software Verify for many years. We recently decided to make it a bit more user friendly and to make it available for public use.

Part



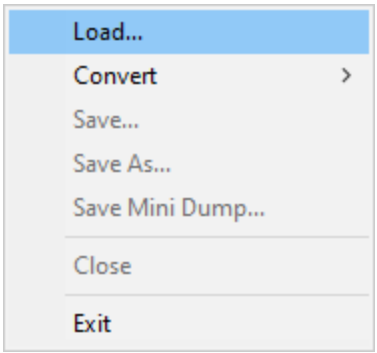
3 Menu

The main menu contains five menus, File, Debug, Settings, Software Updates and Help.



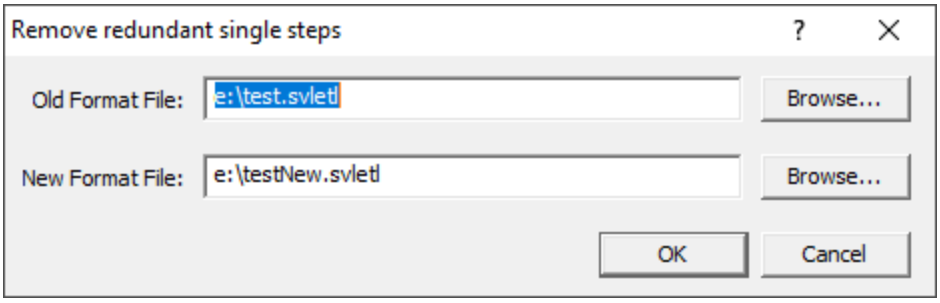
3.1 File

The File menu controls loading and saving of previously saved exception traces.



File menu > **Load...** > loads a previously saved exception trace and displays the data on the user interface.

File menu > **Convert** > **Remove redundant steps from file...** > loads an exception tracer file, removes any redundant steps from the file and then saves the data to a new file.



A redundant single step is a single step that doesn't change the filename and line number.

Removing redundant single steps is very useful for making an unusably large trace full of single steps into a smaller more useful trace.

Although this conversion process is slow and time consuming it is much faster than re-recording the trace with more restrictive single step settings.

File menu > **Save...** > saves the current event data as a new exception trace.

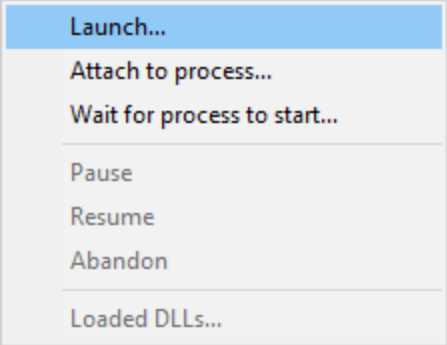
File menu > **Save As...** > saves the current event data as a new exception trace.

File menu > **Close** > clear all results.

File menu > **Exit** > closes Exception Tracer.

3.2 Debug

The Debug menu controls the launching of executables, attaching to running executables and waiting for executables to start.



Debug menu > **Launch...** > displays the launch process dialog.

Debug menu > **Attach to process...** > displays the attach to a running process dialog.

Debug menu > **Wait for process to start...** > displays the wait for a process to start dialog

Debug menu > **Pause** > pause debugging - debug events will not be collected.

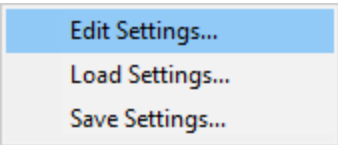
Debug menu > **Resume** > resume debugging - debug events will be collected.

Debug menu > **Abandon** > stop debugging.

Debug menu > **Loaded DLLs...** > displays the list of loaded DLLs.

3.3 Settings

The Settings menu controls editing of the settings controlling Exception Tracer



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

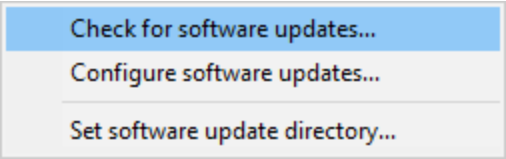
Settings menu > **Load Settings...** > load the settings, choosing load location with the File dialog.

Settings menu > **Save Settings...** > save the settings, choosing save location with the File dialog.

3.4 Software Updates

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

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

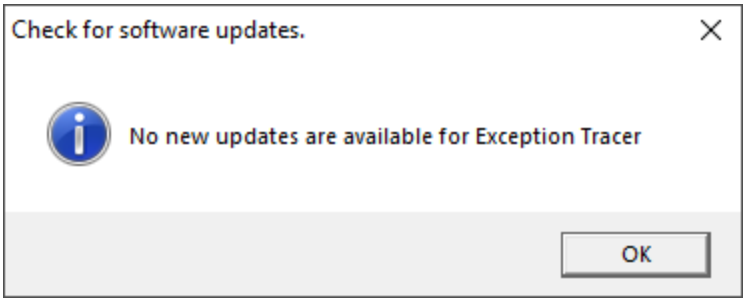


Software Updates menu > **Check for software updates** > checks for updates and shows the software update dialog if any exist

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

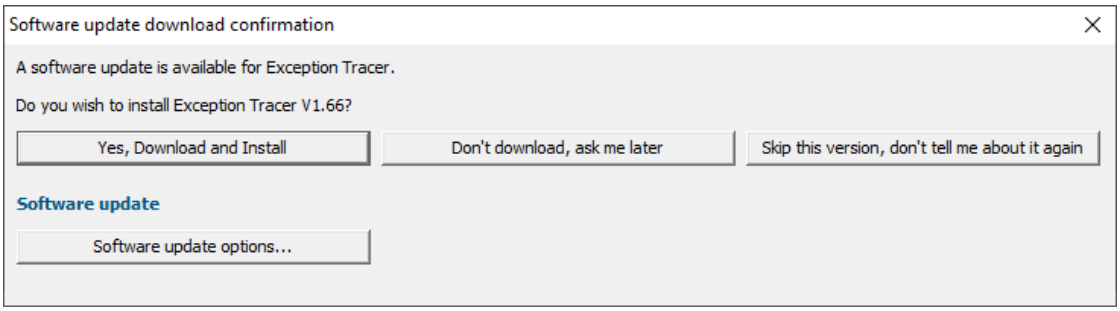
Before updating the software, close the help manual, and end any active session by closing target programs.

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

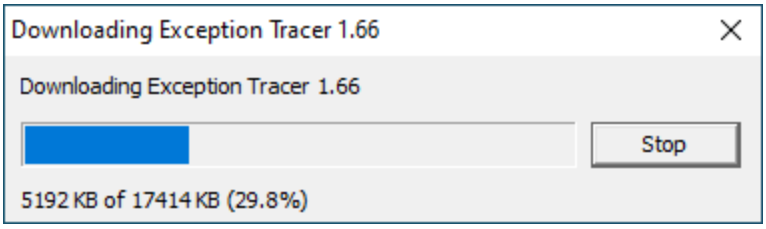


Software Update dialog

If a software update is available for Exception Tracer you'll see the software update dialog.



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



  Once the update has downloaded, Exception Tracer 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 Exception Tracer
- **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 Exception Tracer as follows:

- Ensure Exception Tracer 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

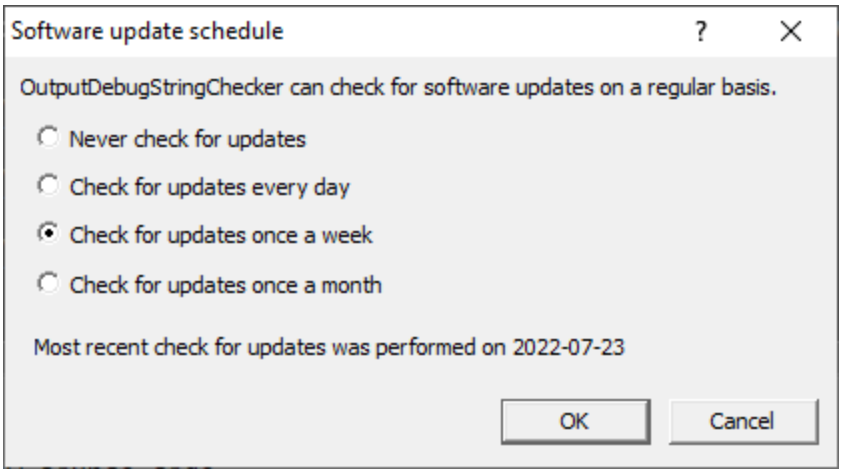
Exception Tracer can automatically check to see if a new version of Exception Tracer is available for downloading.

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.

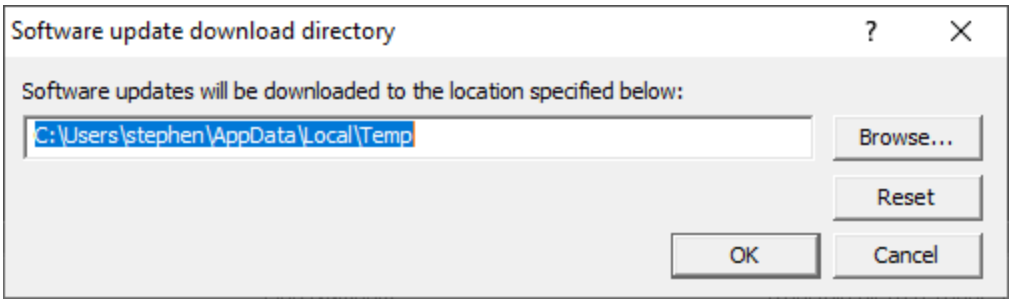


Software update directory

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)

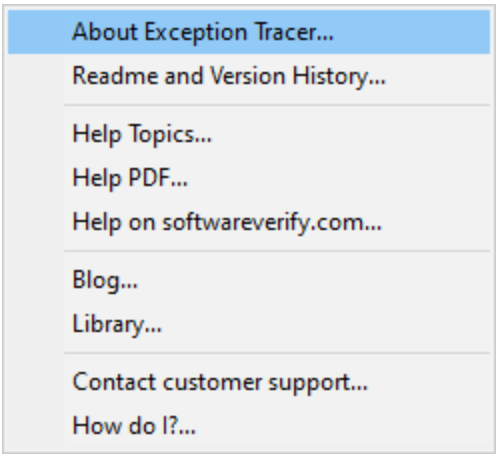
When modifying the download directory, you should ensure the directory will continue to be valid. Updates may no longer occur if the download location is later invalidated.

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

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

3.5 Help

The Help menu controls displaying this help document and displaying information about Exception Tracer.



Help menu > **About Exception Tracer...** > displays information about Exception Tracer.

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

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

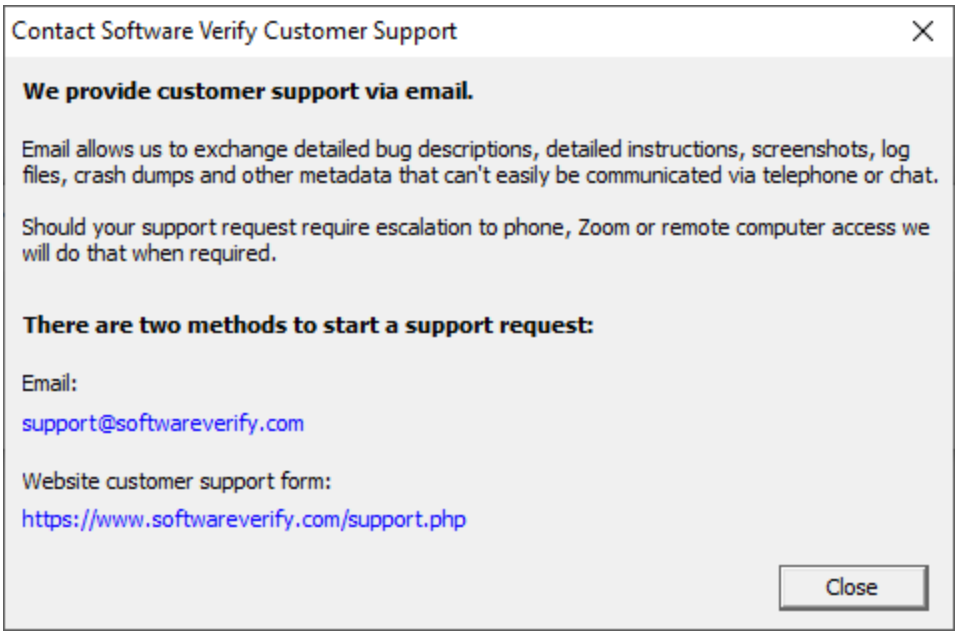
Help menu > **Help PDF...** > displays this help file in PDF format.

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.

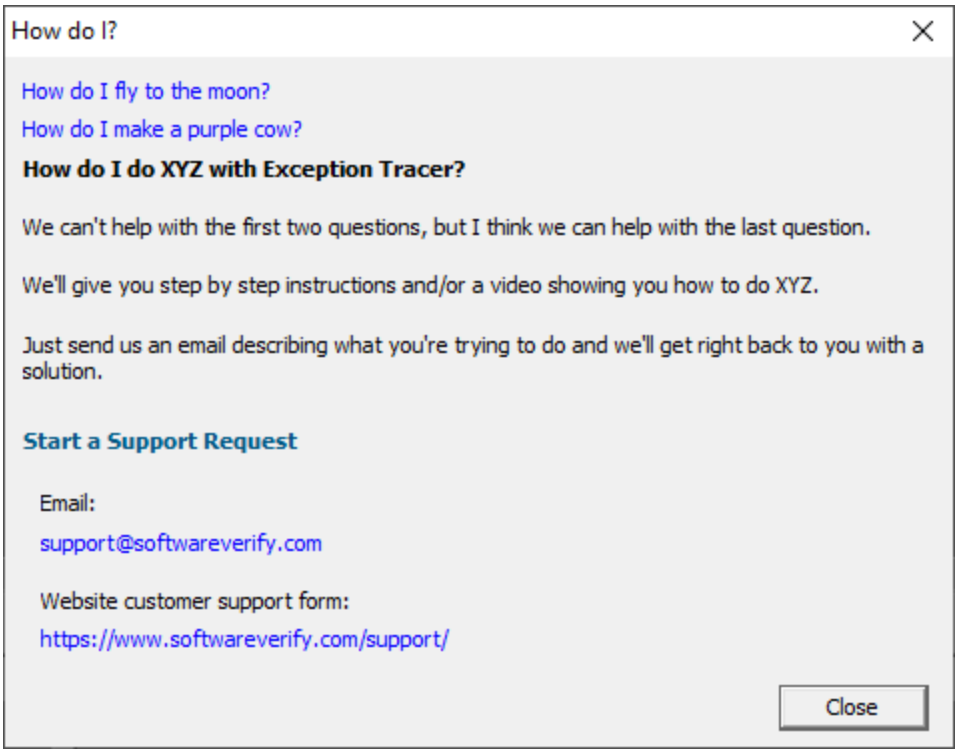
Help menu > **Library...** > display the Software Verify library - our best blog articles grouped by related topics.

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



Click a link to contact customer support.

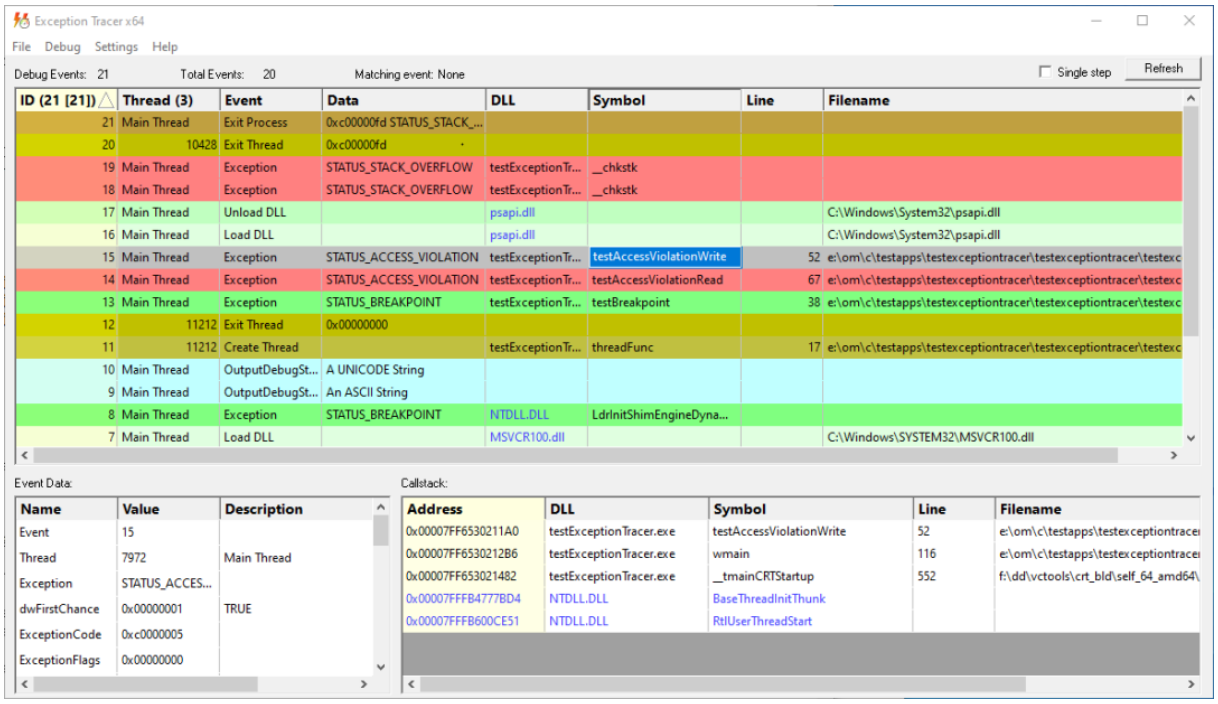
Help menu > **How do I?...** > displays the options for asking us how to do a particular task.



Part IV

4 The user interface

The Exception Tracer user interface is shown below.



The display is split into three areas. The top grid displays debug events, one per line. The bottom two grids display event specific data for the event that is selected in the top grid.

The image above shows six exceptions (the first of which is the breakpoint exception that is always sent when a debugger attaches to a process). One of these exceptions, a write memory access violation, has been selected. The event data and callstack grids show data relating to this write memory access violation.

Single stepping can be enabled or disabled by selecting the Single Step check box. See the single step settings to configure what is collected.

If you have disabled updating of the user interface in the display settings you can manually update the display using the **Refresh** button.

Debug Events

Debug events are displayed in the top grid. Each line in the grid represents one event. Data displayed is the event id, thread name, event type, event data, DLL name, symbol name, line number and filename.

Thread names are sourced from thread naming exceptions and the GetThreadDescription() API (Windows 10 only). Thread names can also be entered via the user interface.

By default the most recent event is displayed first. You can change this from the settings dialog.

Additional sorting can be done by clicking on column headers. Click the column header to sort using that column. Click the same column header to reverse the sort direction.

Not every event will have valid data for each field.

The types of event are:

- **Exception**. Details about an exception. The event data grid shows the exception details. The callstack grid shows the callstack for the exception.
- **Create Thread**. A thread has been created. The thread start function is shown, as are the filename and line number if available.
- **Create Process**. A process has been created. The main thread start function is shown, as are the filename and line number if available.
- **Exit Thread**. A thread has ended. The thread exit code is displayed.
- **Exit Process**. A process has ended. The process exit code is displayed.
- **Load DLL**. A DLL has been loaded.
- **Unload DLL**. A DLL has been unloaded.
- **OutputDebugString**. A debugging string has been output using OutputDebugStringA() or OutputDebugStringW().
- **RIP**. An error reporting event has happened.

Event Data

When an event is selected in the top grid, the event data are displayed in the Event Data grid. The data displayed in the grid vary depending on what type of event has been selected.

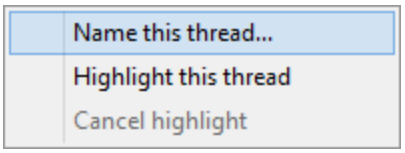
Where possible addresses are converted into symbol names, filenames and line numbers, and error codes and other specific numeric values are converted into strings.

Callstack

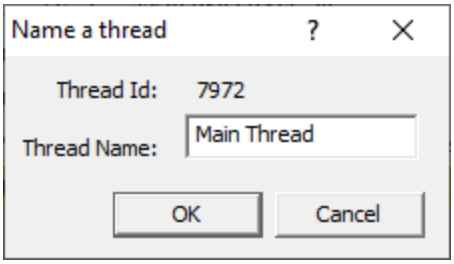
When an event is selected in the top grid, if the event has an associated callstack, the callstack is displayed in the Callstack grid. Where possible addresses are converted into symbol names, filenames and line numbers.

Events Context Menu

The Debug Events grid has a context menu which you can access by right clicking on any event.



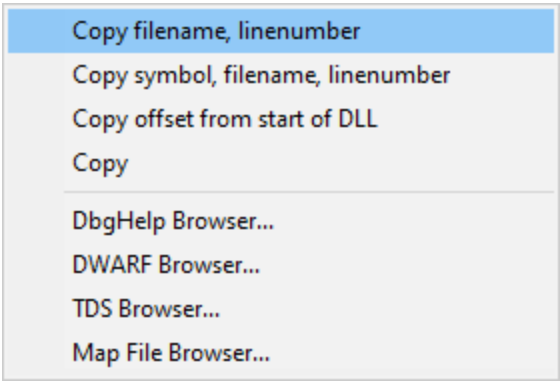
- **Name this thread...** > displays the Name a thread dialog



- **Highlight this thread** > colours all debug events for this thread in the same colour
- **Cancel Highlight** > cancels the highlighting

Callstack Context Menu

The Callstack grid has a context menu which you can access by right clicking on any callstack entry.



- **Copy filename, linenumber** > Copies the filename and line number to the clipboard.
- **Copy symbol, filename, linenumber** > Copies the symbol, filename and line number to the clipboard.
- **Copy offset from start of DLL** > Calculates the stack location as an offset from the start of the DLL and copies it (in hex) to the clipboard.
- **Copy** > Copies the entire line to the clipboard.

- **Exception Tracer** > Exception Tracer is launched to open the PDB symbols associated with the crash DLL. The symbol at the stack location is then highlighted.
- **DWARF Browser** > DWARF Browser is launched to open the DWARF symbols in the crash DLL. The symbol at the stack location is then highlighted.
- **TDS Browser** > TDS Browser is launched to open the TDS symbols associated with the crash DLL. The symbol at the stack location is then highlighted.
- **Map File Browser** > Map File Browser is launched to open the MAP file associated with the crash DLL. The symbol at the stack location is then highlighted.

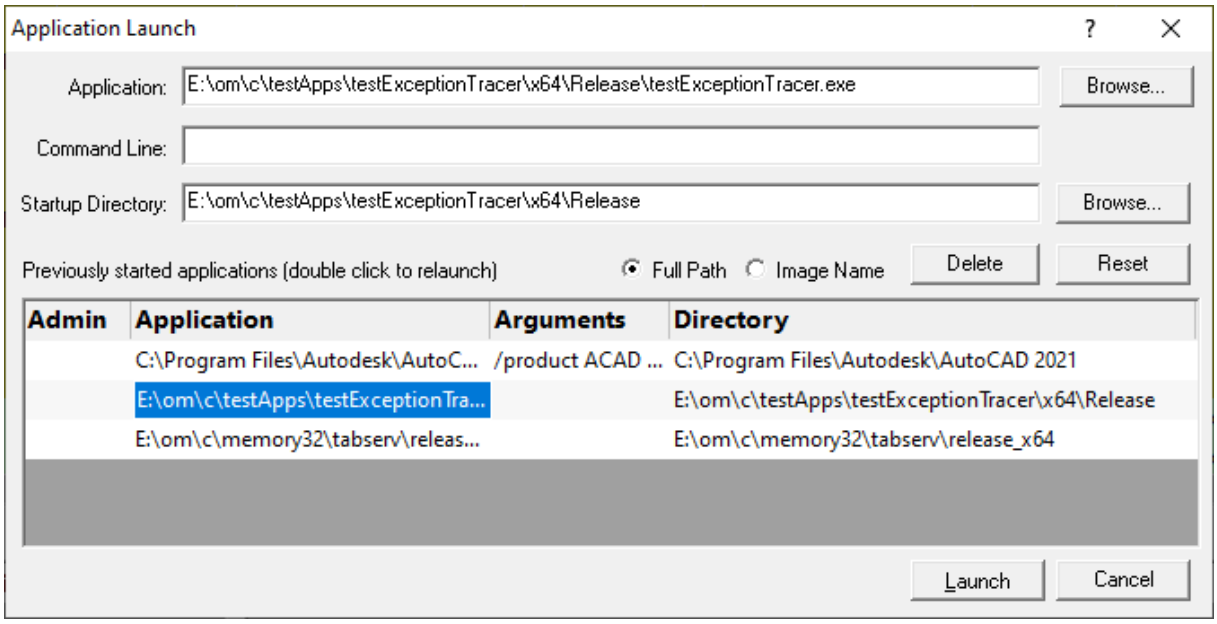
If a tool is chosen that is not installed a prompt will be displayed to download the tool.

Colour Coding

For all three grids, where an event relates to Microsoft DLLs the text for these are colour coded blue to indicate they are different from events for your DLLs.

4.1 Launch dialog

Use the launch dialog to start an application that you wish to monitor for exceptions.



Specifying an executable

To start your debugging session choose your application to start and specify any required command line arguments.

The startup directory will be automatically calculated based on the filename you choose in the Windows File Chooser dialog.

If you type the application filename you'll need to specify the startup directory manually.

Start the debugging session by clicking the **Launch** button.

Relaunching an executable

To relaunch an executable, either select the previously launched application in the grid, the click Launch, or double click the previously launched application.

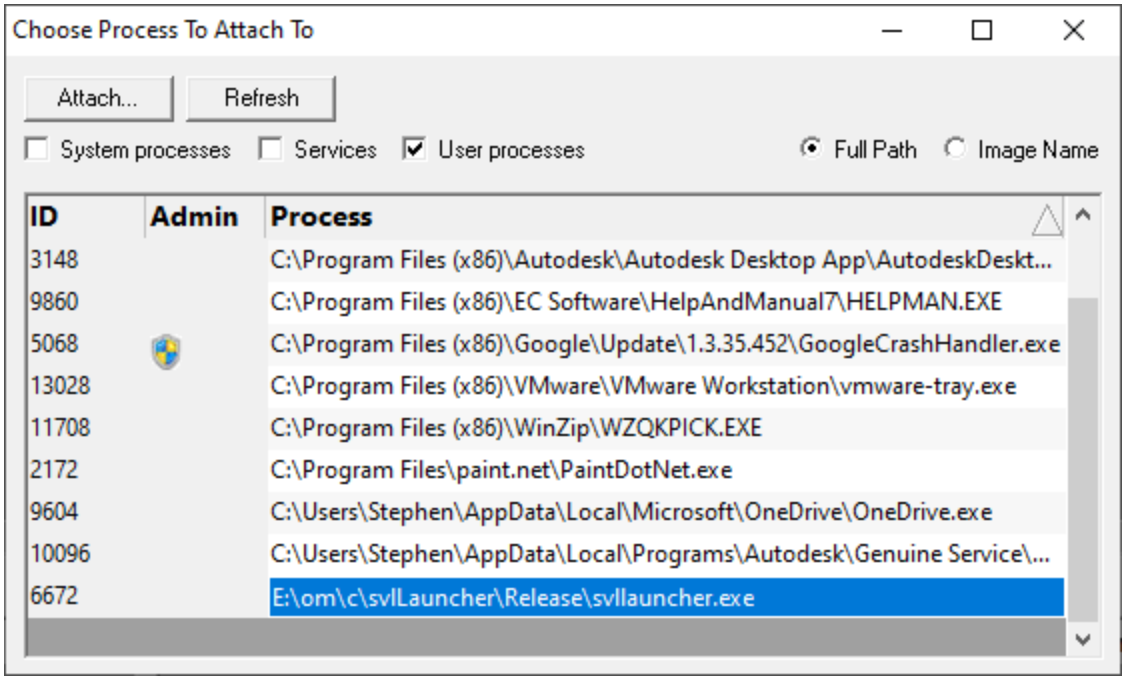
Reset

You can reset the previously launched list of applications with the **Reset** button.

To delete a previously launched application select it them use the Del key or the **Delete** button.

4.2 Attach to process dialog

The Attach to process dialog allows you to choose a user process, system process or service to attach to.

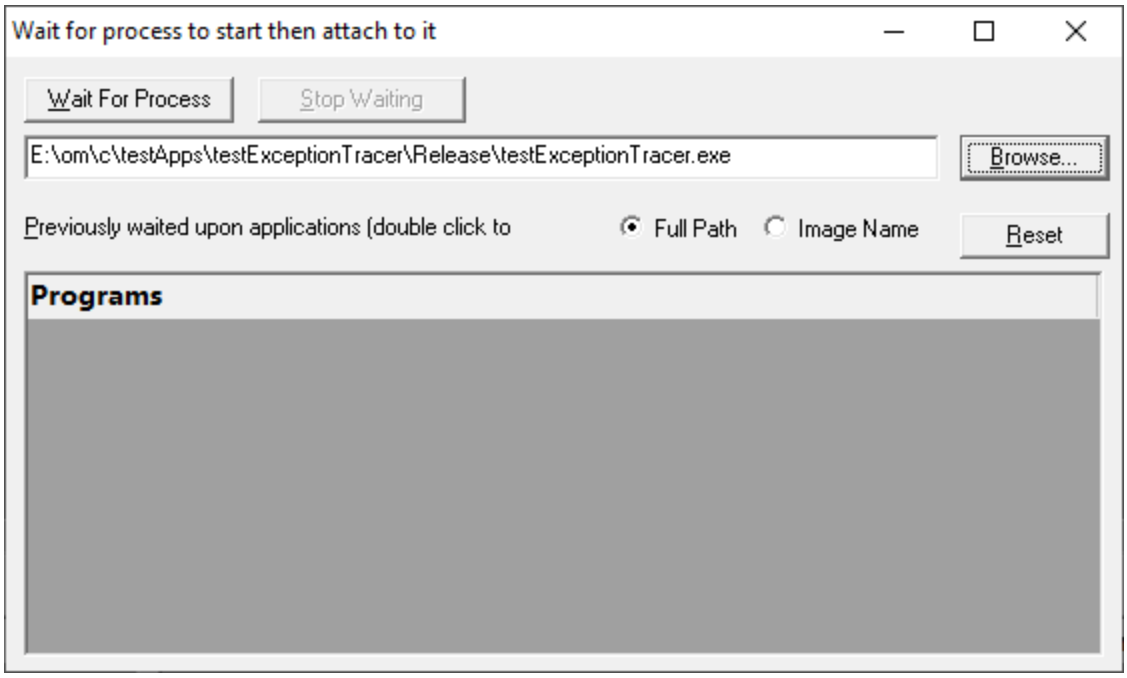


- **System processes / Services / User processes** > show either of system or services or user processes in the list, or both
- **Full path** > shows the full path to the process executable in the list
- **Image Name** > shows the short program name without path
- **Refresh** > update the list with currently running processes
- **Attach** > attach to the selected process and start collecting debug events

Clicking on the headers of the list will sort them by ID or by name using the full name or short name, depending on what's displayed.

4.3 Wait for process dialog

The Wait for process dialog allows you to specify a process that when it starts, Exception Tracer will attach to and start collecting debug events.

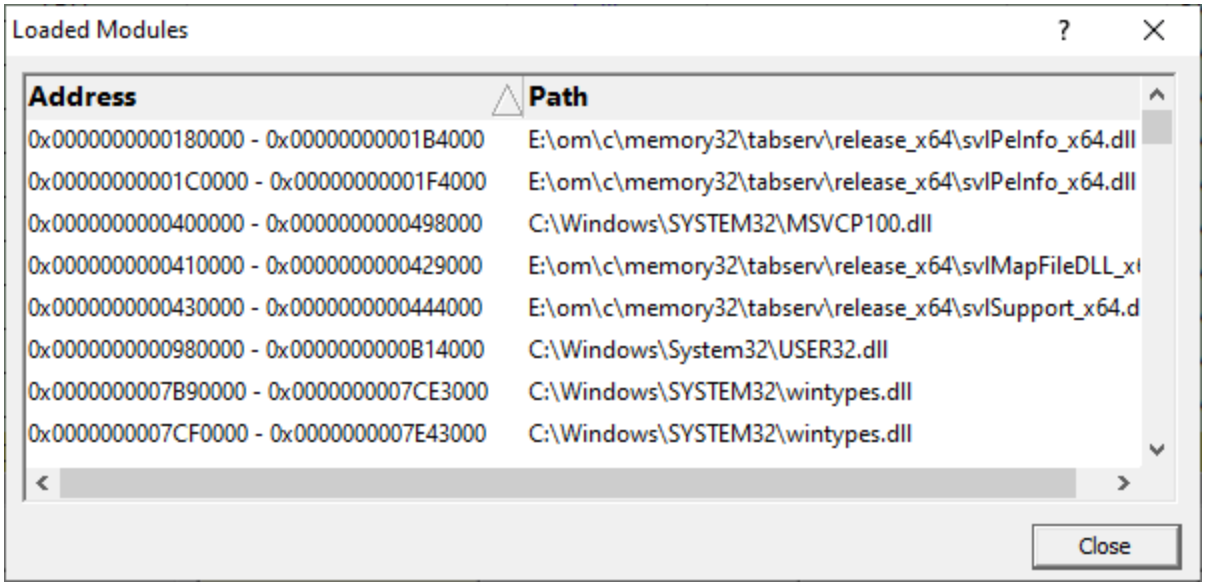


- **Application to wait for** > type or **Browse** to set the application name to launch
- **Full path** > shows the full path to the process executable in the list
- **Image Name** > shows the short program name without path
- **Reset** > clears the list

- **Wait for Process** > start waiting for the process to start. When the process starts Exception Tracer will attach to it and start collecting debug events.
- **Stop Waiting** > stops waiting for the process to start

4.4 Loaded DLLs dialog

The Loaded DLLs dialog displays the list of loaded DLLs.



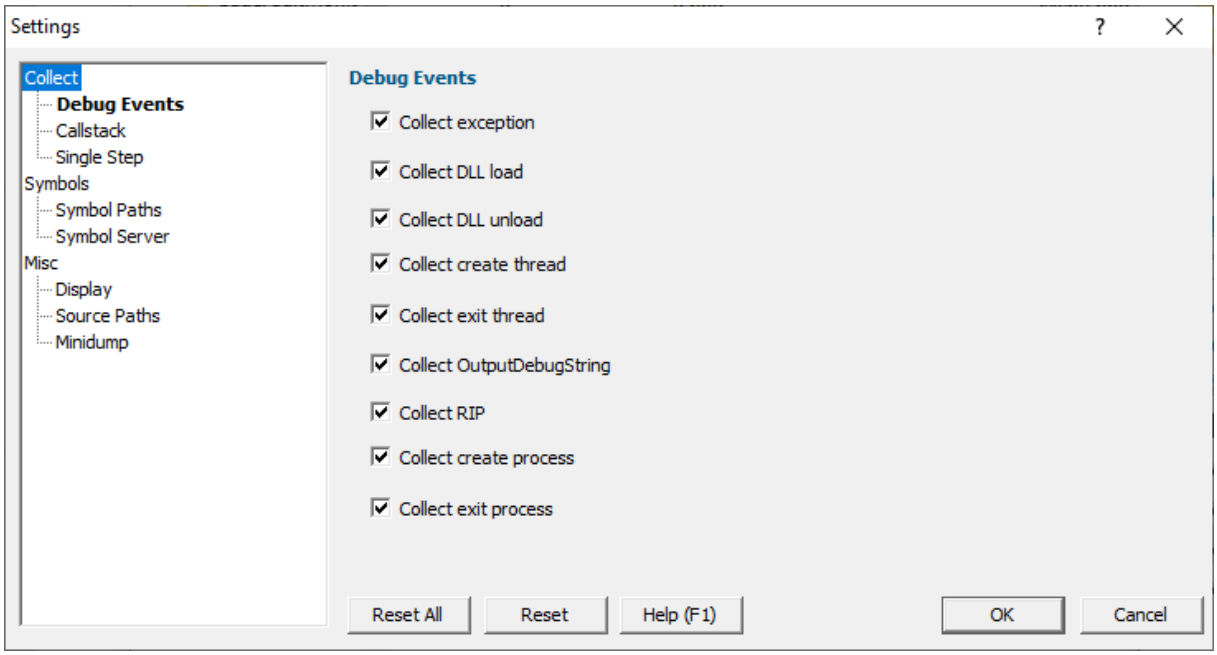
For each DLL the load address and the name of the DLL is displayed.

The data display can be sorted by load address or Path to the DLL.

4.5 Settings dialog

The settings dialog allows you to control the behaviour of Exception Tracer.

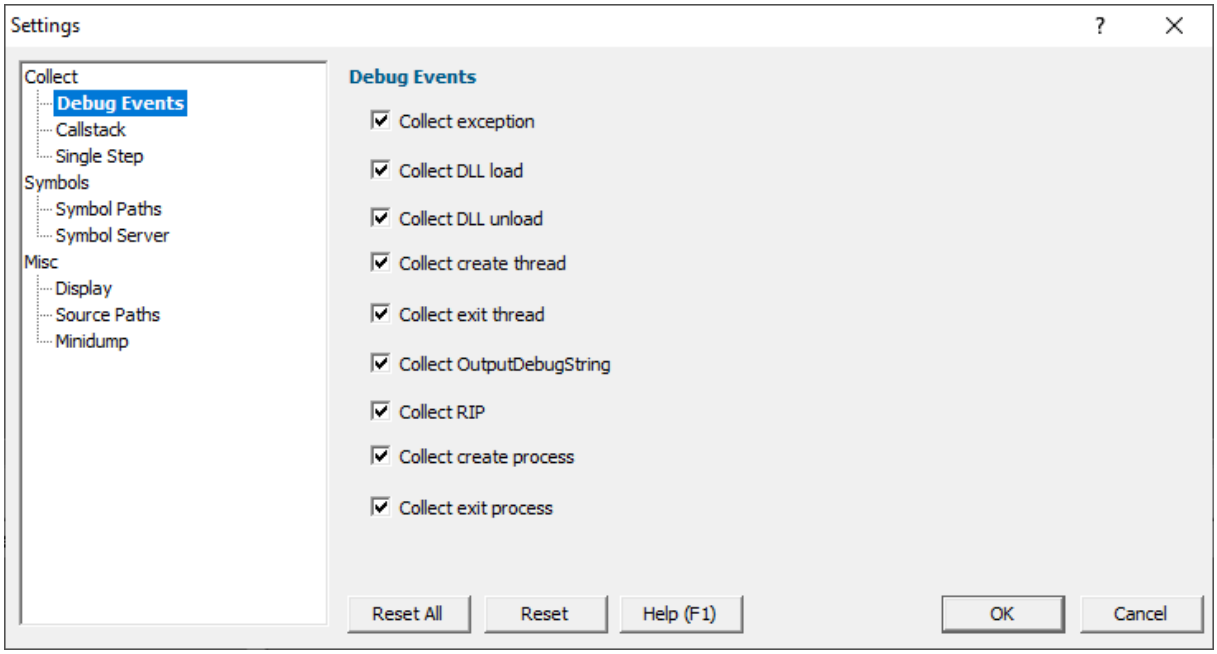
There are three main sections: Collect, Symbols and Misc.



4.5.1 Collect

4.5.1.1 Debug Events

The Debug Events settings allow you to choose which events will be collected by Exception Tracer and which will be ignored.



By default, all options are selected.

Debug Events

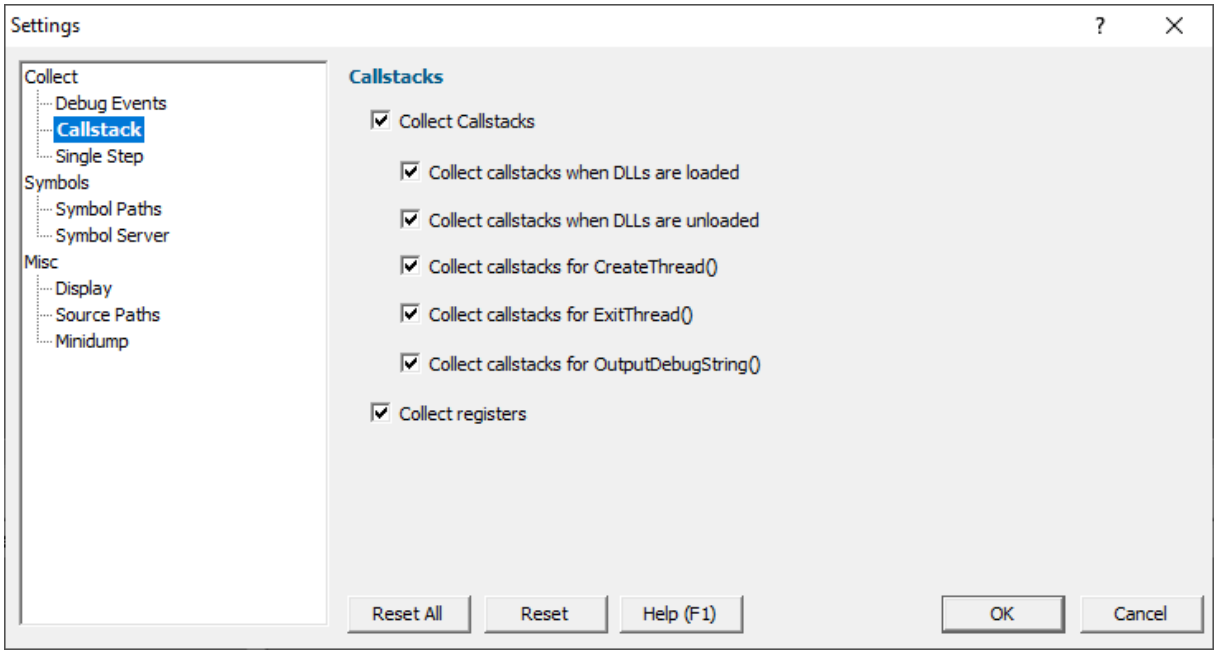
Choose which events are collected.

- **Exception**
- **Load DLL**
- **Unload DLL**
- **Create Thread**
- **Exit Thread**
- **OutputDebugString**
- **RIP**
- **Create Process**
- **Exit Process**

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.1.2 Callstack

The Collect settings allow you to choose which events will be collected by Exception Tracer and which will be ignored.



By default, all options are selected.

Callstacks

Choose which events have a callstack collected.

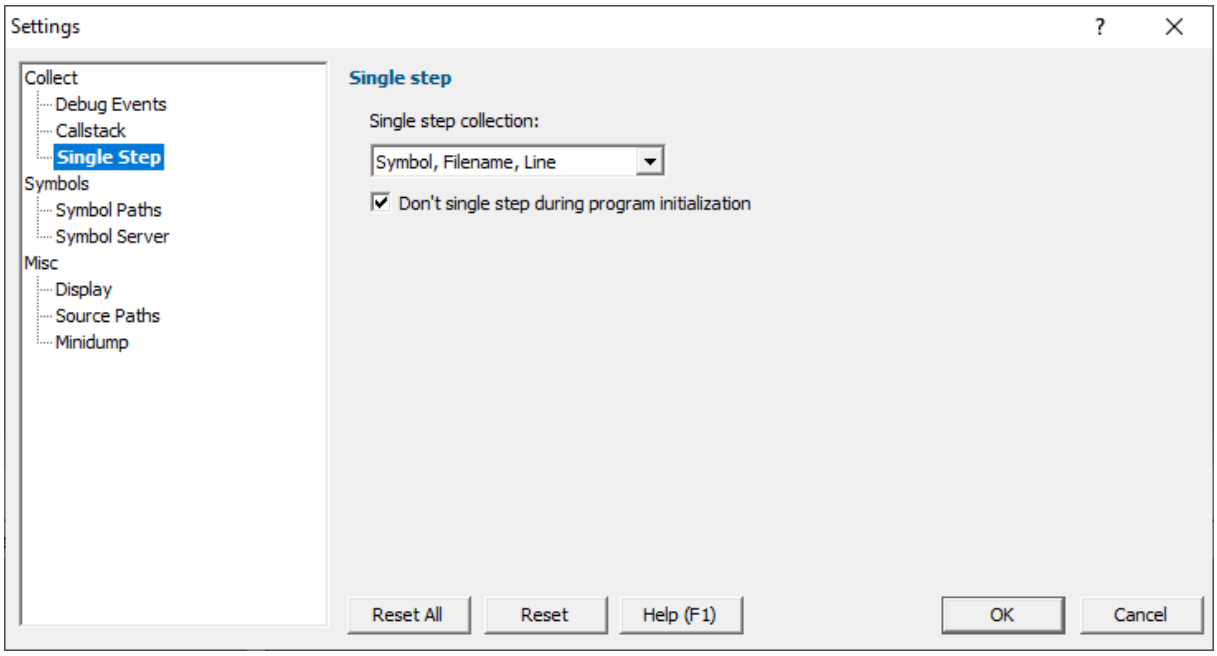
- **Load DLL**
- **Unload DLL**
- **Create Thread**
- **Exit Thread**
- **OutputDebugString**

Callstacks are always collected for **Exception** events.

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.1.3 Single Step

The Collect settings allow you to choose which events will be collected by Exception Tracer and which will be ignored.



By default, single step collection is set to Symbol, Filename, Line.

Single Step Data Collection

Single step data collection results in a lot of data being collected. Most of the time a lot of this data is redundant. We've provided some options to allow you to choose how much data you collect when in single stepping mode

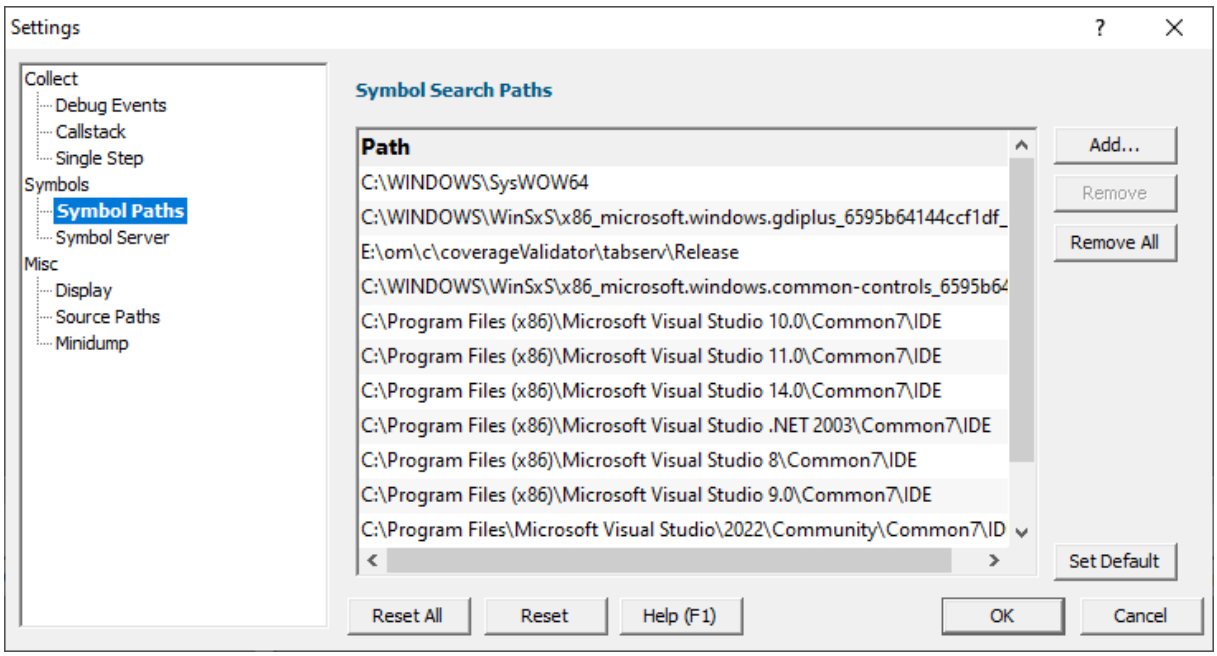
- **All**. All single step events are collected.
- **Symbol, Filename, Line**. Only single step events where either symbol name, filename or line have changed (compared to previous line) are collected.
- **Symbol, Filename**. Only single step events where either symbol name or filename have changed (compared to previous line) are collected.
- **Filename**. Only single step events where filename has changed (compared to previous line) are collected.

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.2 Symbols

4.5.2.1 Symbol Paths

The Symbols settings allow you to specify where Exception Tracer looks for symbols.



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** › adds all valid directories found in the _NT_SYMBOL_PATH environment variable, plus the Windows symbols directory

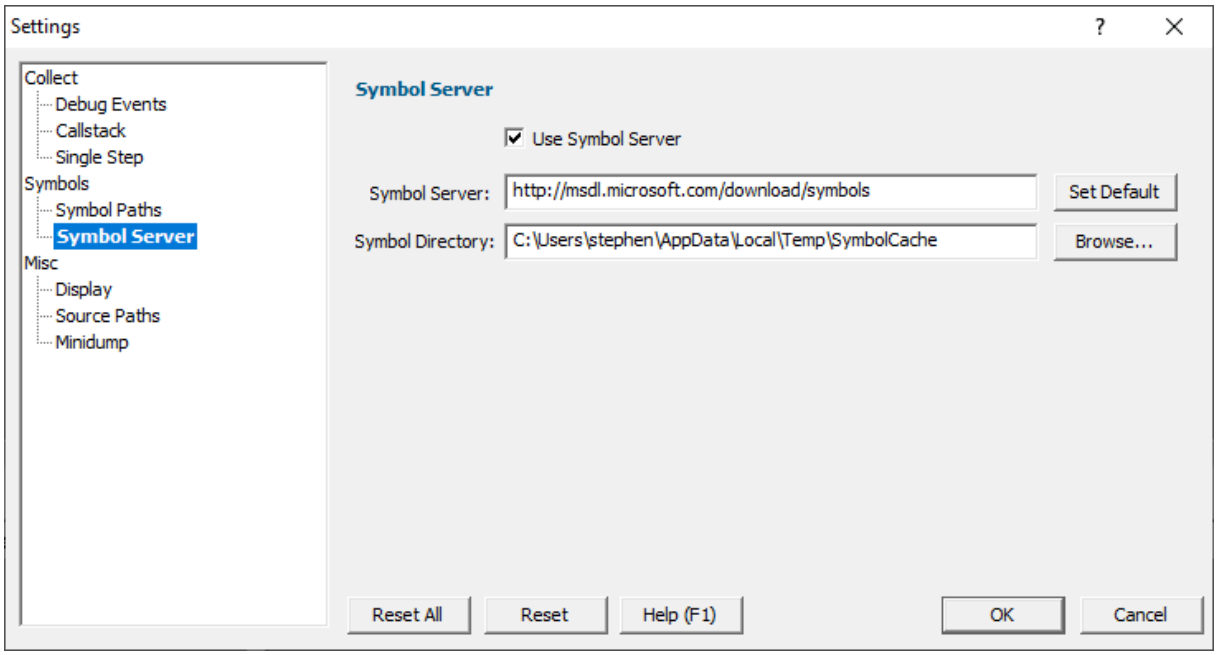
Alternatively, press Del to delete selected items, and Ctrl + A to select all items in the list first.

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.2.2 Symbol Server

The Symbol Server settings allow you to specify what symbol servers to use.

You do not *need* to specify a symbol server if you do not wish to, and Exception Tracer will work correctly without a symbol server.



Symbol server

The symbol server is entirely optional, but is useful for obtaining symbols from a centralized company resource or for obtaining operating symbols from Microsoft.

The default symbol server is the Microsoft symbol server used for acquiring symbols about Microsoft's operating system DLLs. You may also wish to add some symbol servers for any software builds in your organisation.

A symbol server is defined by at least the following:

- the symbol server dll to be used to handle the symbol server interaction
- a directory location where symbol definitions are saved
- the server location - a url

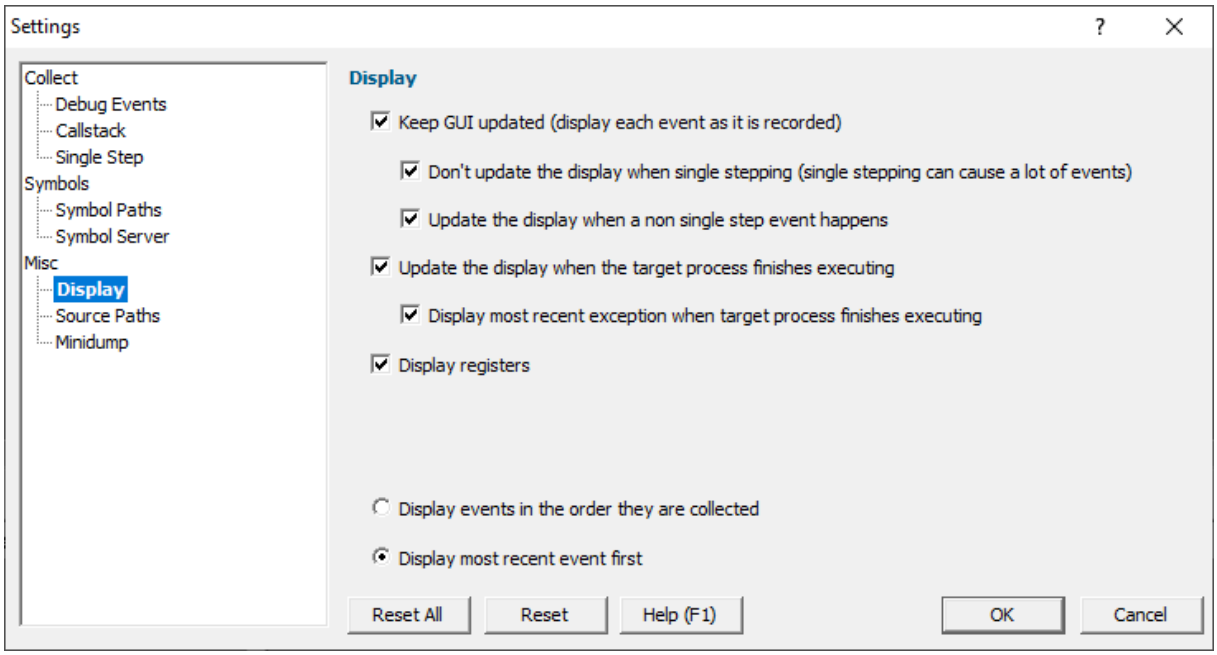
The symbol server can be enabled or disabled allowing you to keep multiple symbol server configurations available without constantly editing their definitions.

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.3 Misc

4.5.3.1 Display

The Display settings allow you to choose how and when data that is collected will be displayed.



Keep GUI updated

Each event will be displayed as it is recorded. For most non-single stepping uses this is what you want.

If you have enabled single stepping a very large amount of data will be collected. Keeping the display up to date will rapidly become very time consuming. We recommend that you don't update the display for each event when you are single stepping.

Update the display when exit process
This ensures that regardless of other options the display is updated at the point of the process finishing execution.

When you're debugging exceptions it's useful if the most recent exception is displayed when the target process finishes executing. This is the default. Turn this off if it's not what you want.

Display registers
If enabled this will display registers in the Event Data part of the user interface. Any values that match already known values (NULL, 0xcdcdcdcd, etc) will be highlighted and if appropriate a readable string will be supplied.

Display Order

While data is being collected the default display order is to show the most recent event first. If you prefer to show events in the order they are collected that option is also available.

Once data collection has ceased you can sort the events by clicking the column headers on the display.

Manual Update

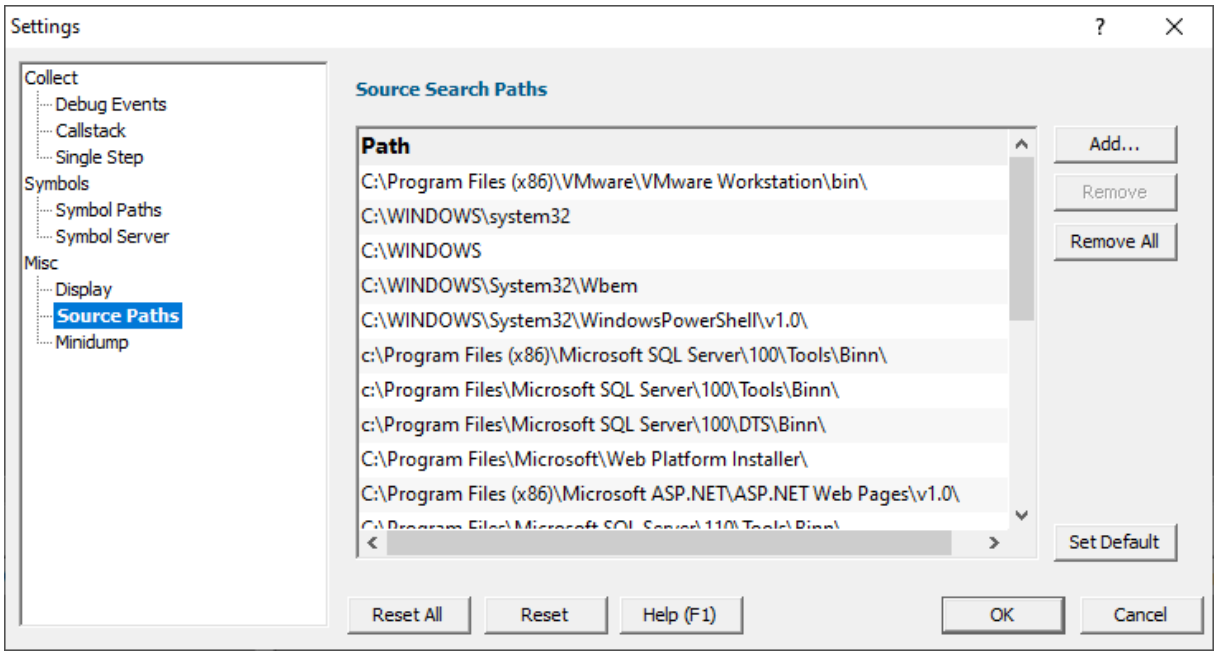
If you have chosen to turn off all data updating you can manually update the main display using the **Refresh** button on the main display.

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.3.2 Source Paths

The Source Paths settings allow you to specify where Exception Tracer looks for source code files.

The source code paths are used when a filename is incomplete - a filename without a path, a filename with a partial path, or a filename that isn't valid on this machine.



Manually adding path type directories

The Path list shows all the paths that will be searched for source code 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:

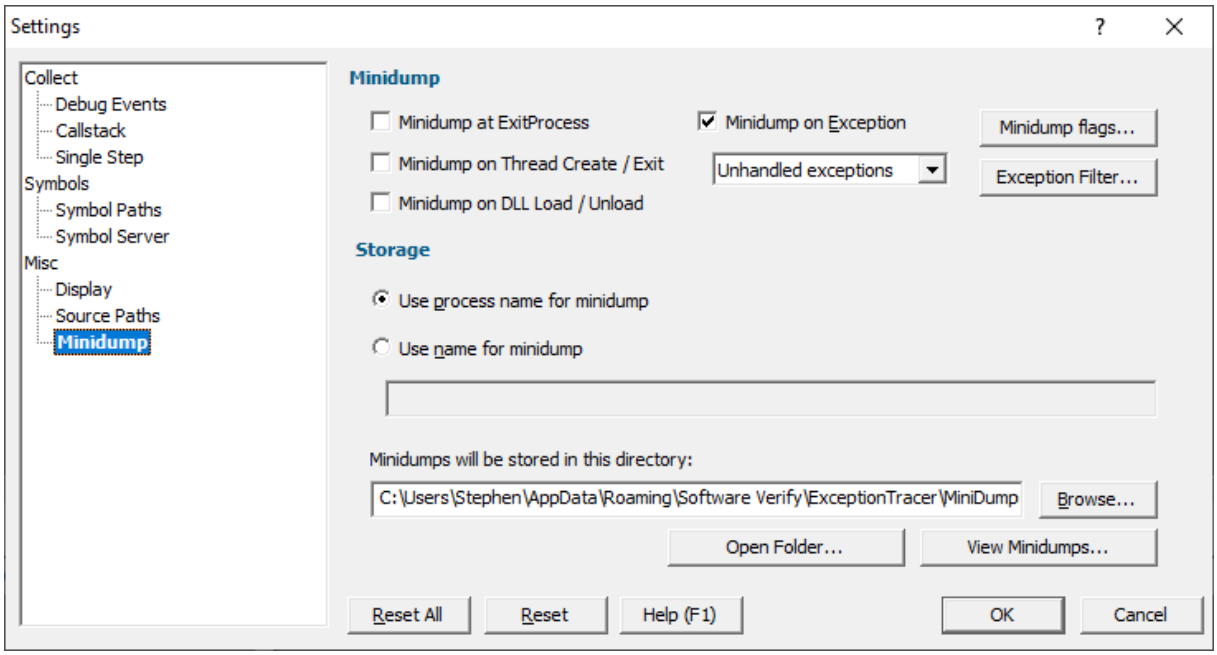
- **Remove** › removes selected items from the list
- **Remove All** › clears the list
- **Set Default** › adds all valid directories found in the PATH environment variable

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

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.3.3 Minidump

The Minidump settings allow you specify if minidumps should be created and where they should be stored.



When to create a Minidump

Minidumps can be created when exceptions are thrown, when threads are created or threads exit, when dlls load and unload, and also when the process exits.

Minidumps can be created containing many different types of information. The default type of Minidump is "Normal", but if you wish to have a different type of mini dump you can change the flags that are used to create a minidump.

Because creating minidumps for every exception type could get a bit cumbersome you can setup exception filters so that you only create a minidump when the exceptions you are interested happen.

If you don't need minidumps, just leave the check boxes cleared.

How to store Minidumps

Minidump storage works by taking a base name, adding a timestamp, and then an exception id. Each exception increases the exception id by 1.

For example:

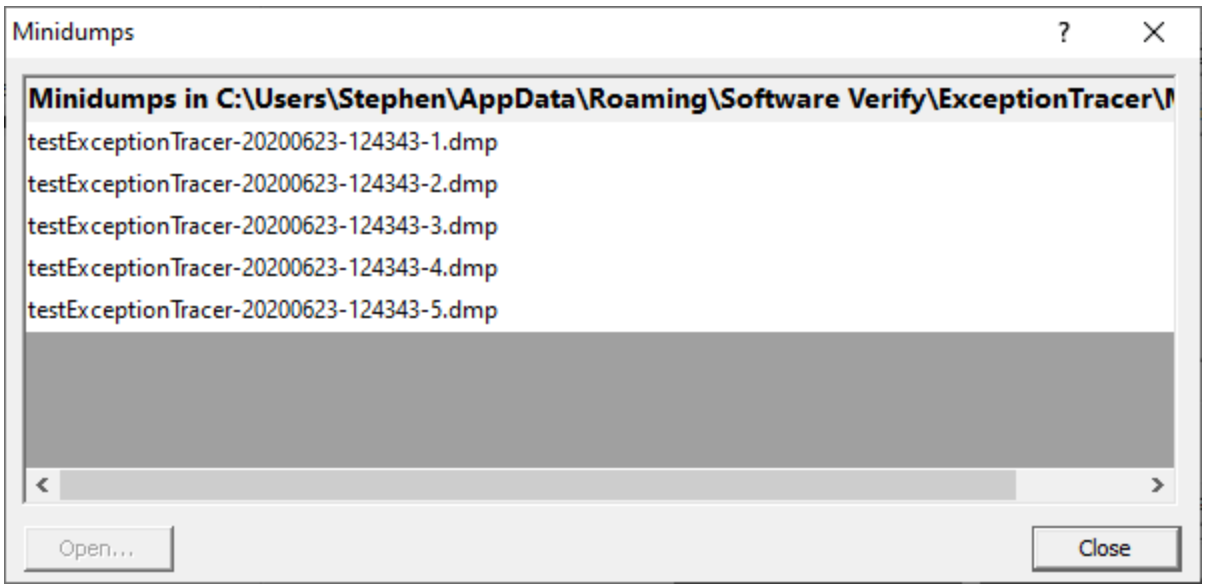
basename-YYYYMMDD-HHMMSS-N.dmp

basename-20190823-125010-1.dmp
basename-20190823-125010-2.dmp
basename-20190823-125010-3.dmp

You can specify your own basename or just choose to use the name of the process you are debugging.

By default we store your minidumps in your AppData folder, but if you'd rather store them somewhere else you can provide your own location.

If you choose to view the list of minidumps you'll be shown the Minidumps dialog.



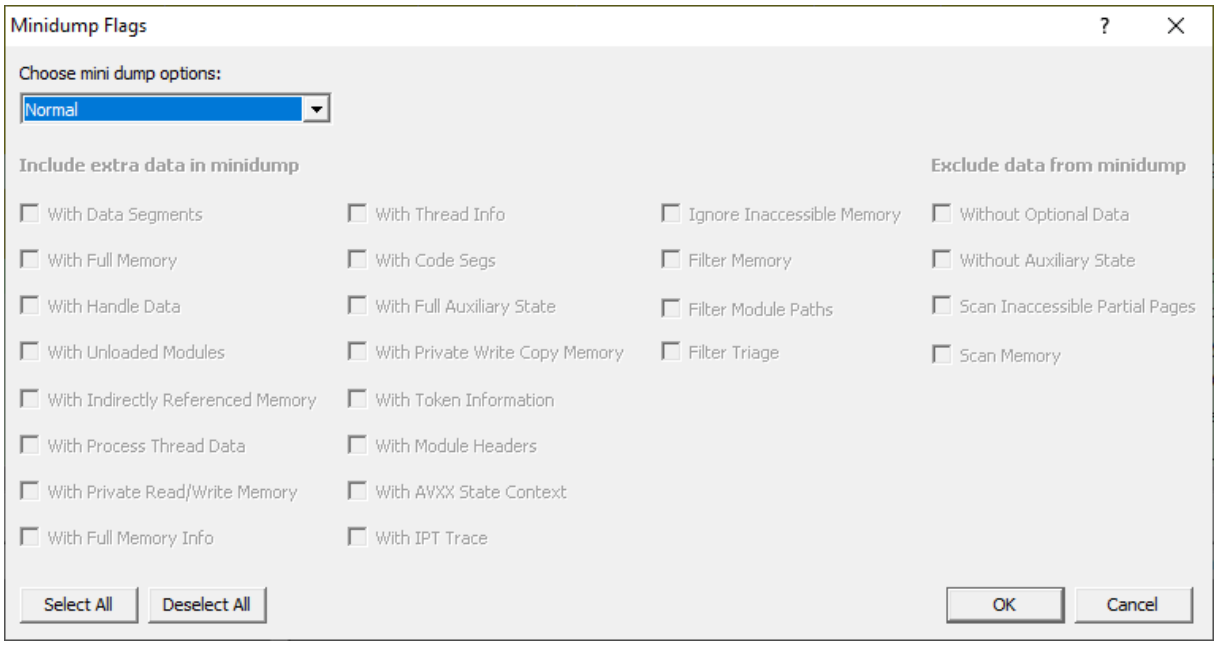
Selecting an entry in the Minidumps dialog will allow you to open it - the minidump will be opened in Visual Studio.

Reset - Resets **all** global settings, not just those on the current page. This includes removing any symbol servers added.

4.5.3.3.1 Minidump Flags

The Minidump Flags dialog allows you to specify what data a minidump contains.

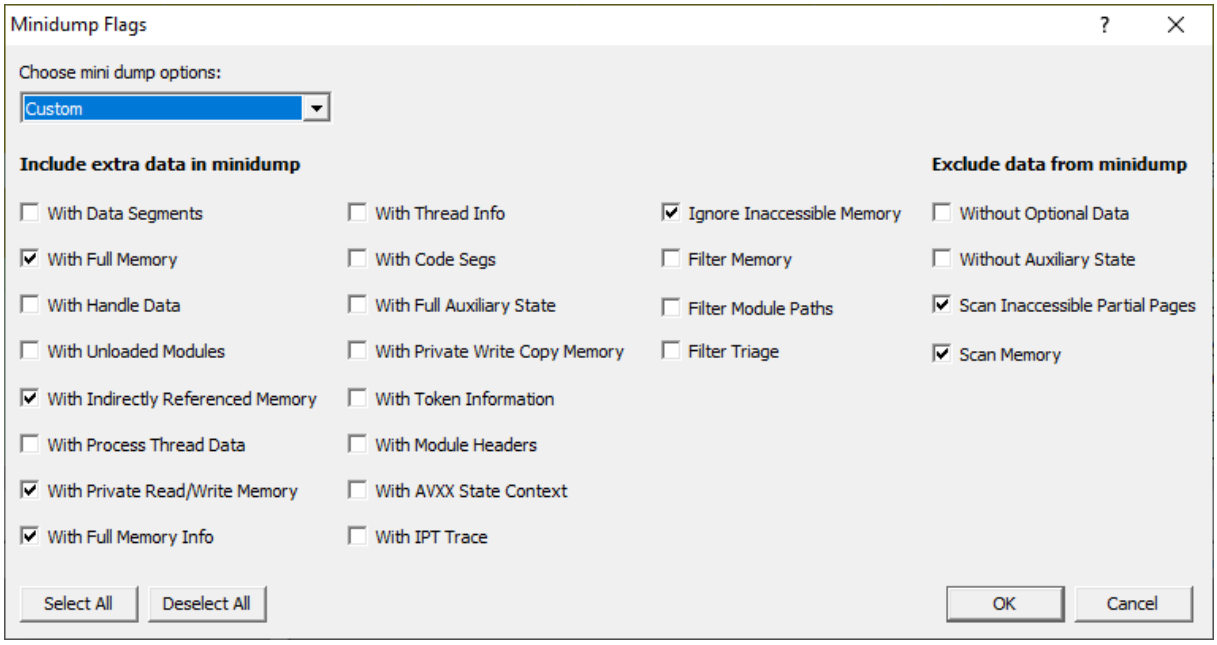
The default options are for a "Normal" minidump.



The default is to generate a "Normal" minidump.

Custom Minidump

Change the combo to "Custom" then select the options that will create the type of minidump that you desire.

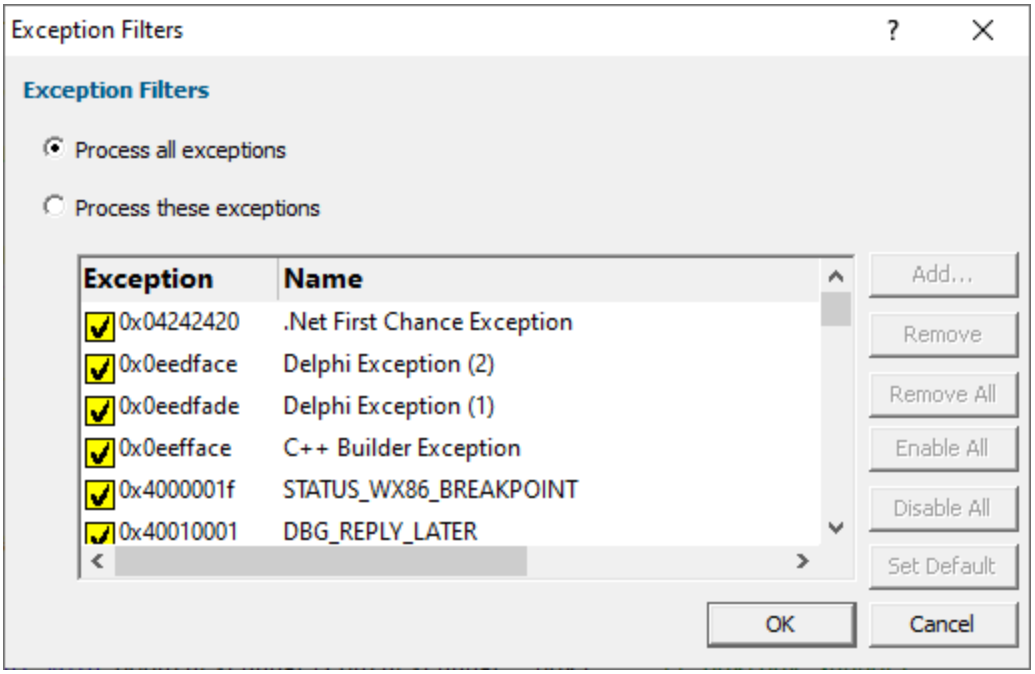


The options in the left hand two columns add data to minidumps, the options in the right hand column restrict data that is in minidumps.

For a detailed explanation of these flags please consult Microsoft's Minidump documentation.

4.5.3.3.2 Exception Filters

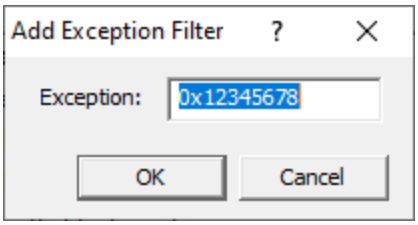
The Exception Filters dialog allows you to specify which exceptions will cause a minidump to be created.



The default is to generate a minidump for all exceptions.

If you wish to only generate minidumps for certain exceptions, select the **Process these exceptions** check box, then choose which exceptions you wish to generate Minidumps for in the list.

If an exception you want isn't listed, click **Add...** to add the exception.



Type the exception number into the dialog.

If you want to reset back to the original exception filters click **Set Default** to set the default exceptions.

Part V

5 Command Line Interface

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

The command line options allow you to launch an application that Exception Tracer will monitor, or to monitor an already running process.

Use in your own debugging work

One usage scenario which we have at Software Verify is to launch Exception Tracer to monitor a worker process we've just launched.

The **/process processId** option is very useful for this, to monitor exceptions in very short lived processes, or to monitor exceptions in the startup phase before you could manually attach with a traditional debugger like Visual Studio.

5.1 Usage Reference

5.1.1 Launch Application

Launch an application that will be monitored by Exception Tracer

/args

Specifies the arguments passed to the process being started

/args arguments to pass to process. Enclose in quotes if any whitespace

Example: /args "-wait 10000 -writeLogOnClose -quantum -blue"

/dir

Specifies the startup directory for the process being started

/dir directory

Example: /dir e:\om\c\

/exe

Specifies the executable that will be started

/exe path-to-executable

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

5.1.2 Monitor Application

Monitor an application that is already running

/process

Specifies the process that will be monitored

/process processId

Example: /process 1344

Monitor an application that is already running

/processName

Specifies the process that will be monitored

/processName process-name.exe

Example: /processName myProcess.exe

Monitor an application that will be started in the future

/waitForProcess

Specifies the process that will be monitored

/waitForProcess process-name.exe

Example: /waitForProcess myProcess.exe

5.1.3 Data Collection

Options to control what data is collected

/collectCallstacks

Collect callstack with each event.

/collectCallstacks:On|Off

Example: /collectCallstacks:On
Example: /collectCallstacks:Off

/collectCallstackLoadDLL

Collect callstack when DLLs are loaded.

/collectCallstackLoadDLL:On|Off

Example: /collectCallstackLoadDLL:On
Example: /collectCallstackLoadDLL:Off

/collectCallstackUnloadDLL

Collect callstack when DLLs are unloaded.

/collectCallstackUnloadDLL:On|Off

Example: /collectCallstackUnloadDLL:On
Example: /collectCallstackUnloadDLL:Off

/collectCallstackCreateThread

Collect callstack when threads are created.

/collectCallstackCreateThread:On|Off

Example: /collectCallstackCreateThread:On
Example: /collectCallstackCreateThread:Off

/collectCallstackExitThread

Collect callstack when threads exit.

/collectCallstackExitThread:On|Off

Example: /collectCallstackExitThread:On
Example: /collectCallstackExitThread:Off

/collectCallstackOutputDebugString

Collect callstack when strings are written to the debugger output with OutputDebugString().

/collectCallstackOutputDebugString:On|Off

Example: /collectCallstackOutputDebugString:On
Example: /collectCallstackOutputDebugString:Off

/collectEventException

Collect data when exceptions are thrown.

/collectEventException:On|Off

Example: /collectEventException:On
Example: /collectEventException:Off

/collectEventCreateThread

Collect data when threads are created.

/collectEventCreateThread:On|Off

Example: /collectEventCreateThread:On
Example: /collectEventCreateThread:Off

/collectEventCreateProcess

Collect data when the process is created.

/collectEventCreateProcess:On|Off

Example: /collectEventCreateProcess:On
Example: /collectEventCreateProcess:Off

/collectEventExitThread

Collect data when threads exit.

/collectEventExitThread:On|Off

Example: /collectEventExitThread:On
Example: /collectEventExitThread:Off

/collectEventExitProcess

Collect data when the process exits.

/collectEventExitProcess:On|Off

Example: /collectEventExitProcess:On
Example: /collectEventExitProcess:Off

/collectEventLoadDLL

Collect data when DLLs are loaded.

/collectEventLoadDLL:On|Off

Example: /collectEventLoadDLL:On
Example: /collectEventLoadDLL:Off

/collectEventUnloadDLL

Collect data when DLLs are unloaded.

/collectEventUnloadDLL:On|Off

Example: /collectEventUnloadDLL:On
Example: /collectEventUnloadDLL:Off

/collectEventOutputDebugString

Collect data when strings are written to the debugger output with OutputDebugString().

/collectEventOutputDebugString:On|Off

Example: /collectEventOutputDebugString:On
Example: /collectEventOutputDebugString:Off

/collectEventRip

Collect data for Rip events.

/collectEventRip:On|Off

Example: /collectEventRip:On
Example: /collectEventRip:Off

/collectRegisters

Collect processor registers with each event.

/collectRegisters:On|Off

Example: /collectRegisters:On
Example: /collectRegisters:Off

5.1.4 Data Display

Options to control what data is displayed

/displayMostRecentEventFirst

Display events with the most recent event first and the oldest event last.

This may seem counterintive at first but it means that all the recent information is at the top of the screen and easily accessible.

The display also updates faster like this because it doesn't need to be scrolled to keep the most recent information in view.

/displayMostRecentEventFirst:On|Off

Example: /displayMostRecentEventFirst:On
Example: /displayMostRecentEventFirst:Off

/displayRegisters

Display registers with each event.

/displayRegisters:On|Off

Example: /displayRegisters:On
Example: /displayRegisters:Off

/keepDisplayUptoDate

Keep the display up to date.

/keepDisplayUptoDate:On|Off

Example: /keepDisplayUptoDate:On
Example: /keepDisplayUptoDate:Off

/dontUpdateTheDisplayWhenSingleStepping

When used with /keepDisplayUptoDate this is a good way of keeping the display up to date without killing performance by listing every single stepping event.

/dontUpdateTheDisplayWhenSingleStepping:On|Off

Example: /dontUpdateTheDisplayWhenSingleStepping:On
Example: /dontUpdateTheDisplayWhenSingleStepping:Off

/updateDisplayWhenNonSingleStepEvent

When used with /keepDisplayUptoDate this is a good way of keeping the display up to date without killing performance by listing every single stepping event.

/updateDisplayWhenNonSingleStepEvent:On|Off

Example: /updateDisplayWhenNonSingleStepEvent:On
Example: /updateDisplayWhenNonSingleStepEvent:Off

/updateDisplayWhenProcessFinishesExecution

Updates the display when the process finishes execution.

/updateDisplayWhenProcessFinishesExecution:On|Off

Example: /updateDisplayWhenProcessFinishesExecution:On
Example: /updateDisplayWhenProcessFinishesExecution:Off

/updateDisplayWhenProcessFinishesExecutionDisplayMostRecentException

Updates the display when the process finishes execution and ensures the most recent exception is viewable.

/updateDisplayWhenProcessFinishesExecutionDisplayMostRecentException:On|Off

Example: /updateDisplayWhenProcessFinishesExecutionDisplayMostRecentException:On
Example: /updateDisplayWhenProcessFinishesExecutionDisplayMostRecentException:Off

5.1.5 Minidumps

Options to control what what minidumps contain and where they are stored

What data to save in a minidump

To save a standard minidump just specify this flag.

/flagMiniDumpNormal

Include just the information necessary to capture stack traces for all existing threads in a process.

/flagMiniDumpNormal

To save a custom minidump specify one or more of these flags.

These flags add data to the minidump.

For a detailed explanation of these flags please consult Microsoft's Minidump documentation.

/flagMiniDumpFilterMemory

Only minimal memory necessary to reconstruct a stack trace.

/flagMiniDumpFilterMemory:On|Off

Example: /flagMiniDumpFilterMemory:On
Example: /flagMiniDumpFilterMemory:Off

/flagMiniDumpFilterModulePaths

Restrict module paths if important information present.

/flagMiniDumpFilterModulePaths:On|Off

Example: /flagMiniDumpFilterModulePaths:On
Example: /flagMiniDumpFilterModulePaths:Off

/flagMiniDumpFilterTriage

Include filter triage data.

/flagMiniDumpFilterTriage:On|Off

Example: /flagMiniDumpFilterTriage:On
Example: /flagMiniDumpFilterTriage:Off

/flagMiniDumpIgnoreInaccessibleMemory

Ignore memory errors when building the minidump.

/flagMiniDumpIgnoreInaccessibleMemory:On|Off

Example: /flagMiniDumpIgnoreInaccessibleMemory:On
Example: /flagMiniDumpIgnoreInaccessibleMemory:Off

/flagMiniDumpWithAvxXStateContext

Include AVX crash state context registers.

/flagMiniDumpWithAvxXStateContext:On|Off

Example: /flagMiniDumpWithAvxXStateContext:On
Example: /flagMiniDumpWithAvxXStateContext:Off

/flagMiniDumpWithCodeSegs

Include the executable code sections from all loaded modules.

/flagMiniDumpWithCodeSegs:On|Off

Example: /flagMiniDumpWithCodeSegs:On
Example: /flagMiniDumpWithCodeSegs:Off

/flagMiniDumpWithDataSegs

Include the data sections from all loaded modules.

/flagMiniDumpWithDataSegs:On|Off

Example: /flagMiniDumpWithDataSegs:On
Example: /flagMiniDumpWithDataSegs:Off

/flagMiniDumpWithFullAuxiliaryState

Include data from 3rd party minidump providers.

/flagMiniDumpWithFullAuxiliaryState:On|Off

Example: /flagMiniDumpWithFullAuxiliaryState:On
Example: /flagMiniDumpWithFullAuxiliaryState:Off

/flagMiniDumpWithFullMemory

Include all accessible memory in the process.

/flagMiniDumpWithFullMemory:On|Off

Example: /flagMiniDumpWithFullMemory:On
Example: /flagMiniDumpWithFullMemory:Off

/flagMiniDumpWithFullMemoryInfo

Include memory region information.

/flagMiniDumpWithFullMemoryInfo:On|Off

Example: /flagMiniDumpWithFullMemoryInfo:On
Example: /flagMiniDumpWithFullMemoryInfo:Off

/flagMiniDumpWithHandleData

Include information about operating system handles that are active.

/flagMiniDumpWithHandleData:On|Off

Example: /flagMiniDumpWithHandleData:On
Example: /flagMiniDumpWithHandleData:Off

/flagMiniDumpWithIndirectlyReferencedMemory

Include pages with data referenced by locals or other stack memory.

/flagMiniDumpWithIndirectlyReferencedMemory:On|Off

Example: /flagMiniDumpWithIndirectlyReferencedMemory:On
Example: /flagMiniDumpWithIndirectlyReferencedMemory:Off

/flagMiniDumpWithIptTrace

Include Intel Processor Trace data

/flagMiniDumpWithIptTrace:On|Off

Example: /flagMiniDumpWithIptTrace:On
Example: /flagMiniDumpWithIptTrace:Off

/flagMiniDumpWithModuleHeaders

Include module header data

/flagMiniDumpWithModuleHeaders:On|Off

Example: /flagMiniDumpWithModuleHeaders:On
Example: /flagMiniDumpWithModuleHeaders:Off

/flagMiniDumpWithPrivateReadWriteMemory

Include memory with PAGE_READWRITE access in the minidump.

/flagMiniDumpWithPrivateReadWriteMemory:On|Off

Example: /flagMiniDumpWithPrivateReadWriteMemory:On
Example: /flagMiniDumpWithPrivateReadWriteMemory:Off

/flagMiniDumpWithPrivateWriteCopyMemory

Include memory with PAGE_WRITECOPY status in the minidump.

/flagMiniDumpWithPrivateWriteCopyMemory:On|Off

Example: /flagMiniDumpWithPrivateWriteCopyMemory:On
Example: /flagMiniDumpWithPrivateWriteCopyMemory:Off

/flagMiniDumpWithProcessThreadData

Include process and thread information in the minidump.

/flagMiniDumpWithProcessThreadData:On|Off

Example: /flagMiniDumpWithProcessThreadData:On
Example: /flagMiniDumpWithProcessThreadData:Off

/flagMiniDumpWithThreadInfo

Include thread state information.

/flagMiniDumpWithThreadInfo:On|Off

Example: /flagMiniDumpWithThreadInfo:On
Example: /flagMiniDumpWithThreadInfo:Off

/flagMiniDumpWithTokenInformation

Include security token information.

/flagMiniDumpWithTokenInformation:On|Off

Example: /flagMiniDumpWithTokenInformation:On
Example: /flagMiniDumpWithTokenInformation:Off

/flagMiniDumpWithUnloadedModules

The list of modules that were recently unloaded, if available.

/flagMiniDumpWithUnloadedModules:On|Off

Example: /flagMiniDumpWithUnloadedModules:On
Example: /flagMiniDumpWithUnloadedModules:Off

These flags prevent data being added the minidump

For a detailed explanation of these flags please consult Microsoft's Minidump documentation.

/flagMiniDumpScanInaccessiblePartialPages

Scan inaccessible partial page data.

/flagMiniDumpScanInaccessiblePartialPages:On|Off

Example: /flagMiniDumpScanInaccessiblePartialPages:On
Example: /flagMiniDumpScanInaccessiblePartialPages:Off

/flagMiniDumpScanMemory

Memory is scanned to see if modules are referenced.

/flagMiniDumpScanMemory:On|Off

Example: /flagMiniDumpScanMemory:On
Example: /flagMiniDumpScanMemory:Off

/flagMiniDumpWithoutAuxiliaryState

Don't include data from 3rd party minidump providers.

/flagMiniDumpWithoutAuxiliaryState:On|Off

Example: /flagMiniDumpWithoutAuxiliaryState:On
Example: /flagMiniDumpWithoutAuxiliaryState:Off

/flagMiniDumpWithoutOptionalData

Minimise the size of the minidump by excluding optional data.

/flagMiniDumpWithoutOptionalData:On|Off

Example: /flagMiniDumpWithoutOptionalData:On
Example: /flagMiniDumpWithoutOptionalData:Off

Where to store a minidump

/miniDumpDir

Specify the directory where minidumps will be written.

/miniDumpDir directory-name

Example: /miniDumpDir e:\my-minidumps
Example: /miniDumpDir "e:\path with spaces in it\my-minidumps"

Minidump Name

Minidumps are named using a root name that is either specified or taken from the process name. The root name then has the date and time appended to it.

An example minidump name might be: "UnitTest442-20231129-113748.dmp" for a minidump created at 11:37:48 on 29 November 2023.

Note that this date and time format makes the minidumps naturally date and time ordered using their name.

The date is also unambiguous (unlike the conflict between DDMMYYYY (least significant date) and MMDDYYY (no significant order) used in USA and a few other countries).

/miniDumpName

Specify the name to use for a minidump.

/miniDumpName mini-dump name.

Example: /miniDumpName "UnitTest48"

/miniDumpNameUseProcessName

When creating a minidump name use the process name without the path and without the file extension.

For example a process named e:\test\mytest.exe would give a minidump name of "mytest".

/miniDumpNameUseProcessName

/miniDumpNameUseSpecificName

When creating a minidump name use the name specified with /miniDumpRootName.

/miniDumpNameUseSpecificName

/miniDumpRootName

Specify the root name of a minidump that is used in conjunction with a specific minidump name.

This option overwrites the name specified with /miniDumpName.

/miniDumpRootName root-name

Example: /miniDumpRootName "Test384"

Triggering a minidump

/miniDumpAtExitProcess

A minidump is written when the process being monitored gets to ExitProcess().

/miniDumpAtExitProcess:On|Off

/miniDumpOnException

A minidump is written when exceptions are thrown.

/miniDumpOnException:On|Off

/miniDumpOnExceptionEveryException

A minidump is written for first chance exceptions.

/miniDumpOnExceptionEveryException

Note that /miniDumpOnException must be set to On and the exception must match the exception filters.

/miniDumpOnExceptionUnhandledExceptions

A minidump is written for second chance exceptions.

/miniDumpOnExceptionUnhandledExceptions

Note that /miniDumpOnException must be set to On and the exception must match the exception filters.

The default behaviour of Exception Tracer is as if /miniDumpOnExceptionUnhandledExceptions has been specified.

/miniDumpOnExceptionAllExceptions

A minidump is written for first chance exceptions and second chance exceptions.

/miniDumpOnExceptionAllExceptions

Note that /miniDumpOnException must be set to On and the exception must match the exception filters.

/miniDumpOnDLLLoadUnload

A minidump is written when DLLs load and when DLLs unload.

/miniDumpOnDLLLoadUnload:On|Off

/miniDumpOnThreadCreateExit

A minidump is written when threads are created and when threads exit.

/miniDumpOnThreadCreateExit:On|Off

Exceptions

/miniDumpAllExceptions

A minidump is written for every exception that is thrown, or minidumps are written if the exception matches the exceptions filter list

/miniDumpAllExceptions:On A minidump is written for every exception thrown

/miniDumpAllExceptions:Off A minidump is written if the exception matches the enabled exceptions in the exception filter list

/miniDumpAddExceptionFilter

Add an exception to the exception filter list. This exception will be marked as enabled.

/miniDumpAddExceptionFilter exception-code

The exception code can be in decimal or hexadecimal (specify with a leading 0x).

Example: /miniDumpAddExceptionFilter 0xC0000005

/miniDumpClearExceptionFilters

The list of exceptions is cleared. Use this option prior to specifying your own list of filters with /miniDumpAddExceptionFilter.

/miniDumpClearExceptionFilters

/miniDumpResetExceptionFilters

The list of exceptions is reset to the standard list of exceptions monitored by Exception Tracer.

/miniDumpResetExceptionFilters

5.1.6 Miscellaneous

Options that don't fit neatly into the other categories

/loadSettings

Load some saved settings from a file.

/loadSettings c:\settings\my-settings.ets

You can save settings from the Settings menu.

/quitWhenProgramFinish

Close Exception Tracer when the target application finishes executing.

/quitWhenProgramFinish

5.1.7 Single Stepping

Options for controlling single stepping

/singleStepRequired

Control if single stepping is turned on or off.

/singleStepRequired:On|Off

Example: /singleStepRequired:On
Example: /singleStepRequired:Off

/singleStepDontSingleStepDuringProgramInit

Control if single stepping is turned on or off during program startup.

Keep this turned off if the bug you are interested in does not manifest during program startup.

/singleStepDontSingleStepDuringProgramInit:On|Off

Example: /singleStepDontSingleStepDuringProgramInit:On
Example: /singleStepDontSingleStepDuringProgramInit:Off

/uniqueSingleSteps

Control how single steps are treated.

/uniqueSingleSteps:None|All|SymbolsFilenamesLines|SymbolsFilenames|Filenames

Example: /uniqueSingleSteps:None
Example: /uniqueSingleSteps:All
Example: /uniqueSingleSteps:SymbolsFilenamesLines
Example: /uniqueSingleSteps:SymbolsFilenames
Example: /uniqueSingleSteps:Filenames

See the topic Single Stepping for more information.

5.1.8 Source Paths

Options for controlling where source files can be found

/clearSourcePaths

Clear all source paths.

/clearSourcePaths

/addSourcePath

Add a source paths to the list of places to look for source code.

/addSourcePath filename

5.1.9 Symbol Paths

Options for controlling where symbols can be found

/enableSymbolPaths

Enable the use of symbol paths.

/enableSymbolPaths:On|Off

Example: /enableSymbolPaths:On
Example: /enableSymbolPaths:Off

/clearSymbolPaths

Clear all symbol paths.

/clearSymbolPaths

/addSymbolPath

Add a path to the list of places that will be searched for symbols.

/addSymbolPath filename

Example: /addSymbolPath e:\my-symbols
Example: /addSymbolPath "e:\a directory path containing spaces\my-symbols"

5.1.10 Symbol Servers

Options for setting up a symbol server

/enableSymbolServer

Enable the use of the symbol server.

/enableSymbolServer:On|Off

Example: /enableSymbolServer:On
Example: /enableSymbolServer:Off

/symbolServer

Specify the symbol server to use

/symbolServer symbol-server

Example: /symbolServer http://msdl.microsoft.com/download/symbols

/symbolDirectory

Specify the directory to use to store symbols downloaded from the symbol server.

/symbolDirectory directory

Example: /symbolDirectory C:\Users\JoeBloggs\AppData\Local\Temp\SymbolCache

5.2 Command Line Reference

/args	Specifies the arguments passed to the process being started
/addSourcePath	Add a source paths to the list of places to look for source code
/addSymbolPath	Add a path to the list of places that will be searched for symbols
/clearSourcePaths	Clear all source paths
/clearSymbolPaths	Clear all symbol paths
/collectCallstacks	Collect callstack with each event
/collectCallstackCreateThread	Collect callstack when threads are created
/collectCallstackExitThread	Collect callstack when threads exit
/collectCallstackLoadDLL	Collect callstack when DLLs are loaded
/collectCallstackOutputDebugString	Collect callstack when strings are written to the debugger output with OutputDebugString()
/collectCallstackUnloadDLL	Collect callstack when DLLs are unloaded
/collectEventCreateProcess	Collect data when the process is created
/collectEventCreateThread	Collect data when threads are created
/collectEventException	Collect data when exceptions are thrown
/collectEventExitProcess	Collect data when the process exits
/collectEventExitThread	Collect data when threads exit
/collectEventLoadDLL	Collect data when DLLs are loaded
/collectEventOutputDebugString	Collect data when strings are written to the debugger output with OutputDebugString()
/collectEventUnloadDLL	Collect data when DLLs are unloaded
/collectEventRip	Collect data for Rip events
/collectRegisters	Collect processor registers with each event
/dir	Specifies the startup directory for the process being started
/displayRegisters	Display registers with each event
/dontUpdateTheDisplayWhenSingleStepping	Prevent single stepping events from killing the user interface performance.
/displayMostRecentEventFirst	Display events with the most recent event first and the oldest event last
/enableSymbolPaths	Enable looking in the symbol path directories for symbols
/enableSymbolServer	Enable using the symbol server to lookup symbols
/exe	Specifies the executable that will be started
/flagMiniDumpFilterMemory	Only minimal memory necessary to reconstruct a stack trace
/flagMiniDumpFilterModulePaths	Restrict module paths if important information present
/flagMiniDumpFilterTriage	Include filter triage data
/flagMiniDumpIgnoreInaccessibleMemory	Ignore memory errors when building the minidump
/flagMiniDumpNormal	Include just the information necessary to capture stack traces for all existing threads in a process
/flagMiniDumpScanInaccessiblePartialPages	Scan inaccessible partial page data
/flagMiniDumpScanMemory	Memory is scanned to see if modules are referenced
/flagMiniDumpWithAvxXStateContext	Include AVX crash state context registers

/flagMiniDumpWithCodeSegs	Include the executable code sections from all loaded modules
/flagMiniDumpWithDataSegs	Include the data sections from all loaded modules
/flagMiniDumpWithFullAuxiliaryState	Include data from 3rd party minidump providers
/flagMiniDumpWithFullMemory	Include all accessible memory in the process
/flagMiniDumpWithFullMemoryInfo	Include memory region information
/flagMiniDumpWithHandleData	Include information about operating system handles that are active
/flagMiniDumpWithIndirectlyReferencedMemory	Include pages with data referenced by locals or other stack memory
/flagMiniDumpWithIptTrace	Include Intel Processor Trace data
/flagMiniDumpWithModuleHeaders	Include module header data
/flagMiniDumpWithProcessThreadData	Include process and thread information in the minidump
/flagMiniDumpWithPrivateReadWriteMemory	Include memory with PAGE_READWRITE access in the minidump
/flagMiniDumpWithPrivateWriteCopyMemory	Include memory with PAGE_WRITECOPY status in the minidump
/flagMiniDumpWithThreadInfo	Include thread state information
/flagMiniDumpWithTokenInformation	Include security token information
/flagMiniDumpWithUnloadedModules	The list of modules that were recently unloaded, if available
/flagMiniDumpWithoutAuxiliaryState	Don't include data from 3rd party minidump providers
/flagMiniDumpWithoutOptionalData	Minimise the size of the minidump by excluding optional data
/keepDisplayUptoDate	Keep the display up to date
/loadSettings	Load some saved settings from a file
/miniDumpAddExceptionFilter	Add an exception to the exception filter list
/miniDumpAllExceptions	A minidump is written for every exception that is thrown, or minidumps are written if the exception matches the exceptions filter list
/miniDumpAtExitProcess	A minidump is written when the process being monitored gets to ExitProcess()
/miniDumpClearExceptionFilters	The list of exceptions is cleared. Use this option prior to specifying your own list of filters with /miniDumpAddExceptionFilter
/miniDumpDir	Specify the directory where minidumps will be written
/miniDumpName	Specify the name to use for a minidump
/miniDumpNameUseProcessName	When creating a minidump name use the process name without the path and without the file extension
/miniDumpNameUseSpecificName	When creating a minidump name use the name specified with /miniDumpRootName
/miniDumpOnDLLLoadUnload	A minidump is written when DLLs load and when DLLs unload
/miniDumpOnException	A minidump is written when exceptions are thrown
/miniDumpOnExceptionAllExceptions	A minidump is written for first chance exceptions and second chance exceptions
/miniDumpOnExceptionEveryException	A minidump is written for first chance exceptions
/miniDumpOnExceptionUnhandledExceptions	A minidump is written for second chance exceptions
/miniDumpOnThreadCreateExit	A minidump is written when threads are created and when threads exit
/miniDumpResetExceptionFilters	The list of exceptions is reset to the standard list of exceptions monitored by Exception Tracer

/miniDumpRootName	Specify the root name of a minidump that is used in conjunction with a specific minidump name
/process	Specifies the process that will be monitored
/processName	Specifies the process that will be monitored
/quitWhenProgramFinish	Close Exception Tracer when the target application finishes executing
/singleStepRequired	Control if single stepping is turned on or off
/singleStepDontSingleStepDuringProgramInit	Control if single stepping is turned on or off during program startup
/symbolServer	Specify the symbol server to use
/symbolDirectory	Specify the directory to use to store symbols downloaded from the symbol server
/updateDisplayWhenNonSingleStepEvent	When used with /keepDisplayUptoDate this is a good way of keeping the display up to date without killing performance by listing every single stepping event.
/updateDisplayWhenProcessFinishesExecution	Updates the display when the process finishes execution.
/updateDisplayWhenProcessFinishesExecutionDisplayMostRecentException	Updates the display when the process finishes execution and ensures the most recent exception is viewable
/uniqueSingleSteps	Control how single steps are treated
/waitForProcess	Specifies the process that will be monitored

5.3 Command Line Examples

This section provides some example command lines. For each example we'll break it down, argument by argument so that you can see how the command line works.

In all these examples the executable to run is **exceptionTracer.exe**. You will need to provide the full path to exceptionTracer.exe, or add the path to exceptionTracer install directory to your $PATH.

32 bit application monitoring: **exceptionTracer.exe**
64 bit application monitoring: **exceptionTracer_x64.exe**

Example 1

exceptionTracer.exe /exe e:\om\c\test\release\test.exe /dir e:\om\c\test\release /args "-add 3 4"

/exe e:\om\c\test\release\test.exe

Start application e:\om\c\test\release\test.exe

/dir e:\om\c\test\release

Start the application in e:\om\c\test\release

/args "-add 3 4"

Pass the arguments -add 3 4 to the application being launched

Example 2

exceptionTracer.exe /process 1344

/process 1344

Attach to process 1344 and monitor that process.

Example 3

exceptionTracer.exe /processName myprocess.exe

/processName myprocess.exe

Attach to process myprocess.exe and monitor that process.

Caution: *If there are multiple processes name myprocess.exe the first one found when enumerating the process list is the one that is used. For situations like this we recommend using /process instead.*

Example 4

Launching ExceptionTracer from your own code to monitor a process you've just started. The arguments are the same as for Example 2.

```
int attachExceptionTracerToRunningProcess(DWORD         processId,          // process id to monit
                                          const TCHAR   *dir)               // startup dir, can be
{
   CString      exceptionTracer;
   int          bRet;

   exceptionTracer = getExceptionTracerPath(dir);
   if (GetFileAttributes(exceptionTracer) != INVALID_FILE_ATTRIBUTES)
   {
      // only try to launch if exception tracer is a valid filename

      TCHAR               commandLine[1000];
      STARTUPINFO         stStartInfo;
      PROCESS_INFORMATION stProcessInfo;

      memset(&stStartInfo, 0, sizeof(STARTUPINFO));
      memset(&stProcessInfo, 0, sizeof(PROCESS_INFORMATION));

      stStartInfo.cb = sizeof(STARTUPINFO);

      _stprintf_s(commandLine, sizeof(commandLine) / sizeof(commandLine[0]), _T("%s /process %u"),

      bRet = CreateProcess(NULL,
                           commandLine,
                           NULL,
                           NULL,
                           FALSE,
                           0,
                           NULL,
                           dir,
                           &stStartInfo,
                           &stProcessInfo);
      if (bRet)
      {
         WaitForInputIdle(stProcessInfo.hProcess, 10 * 1000);   // 10 seconds

         CloseHandle(stProcessInfo.hProcess);
         CloseHandle(stProcessInfo.hThread);
      }
   }
   else
      bRet = FALSE;

   return bRet;
}
```

Example 5

exceptionTracer.exe /waitForProcess myprocess.exe

/waitForProcess myprocess.exe

Wait for myprocess.exe to start then attach to myprocess.exe and monitor that process.

Caution: *If a process called myprocess.exe is already running exceptionTracer will monitor that process.*