Help Contents

Click a topic listed below to see information about that topic. Read **Essentials** if you are new to WinSpector or new to this version of WinSpector. If you need information on using the Help system, return to the Help menu and choose Using Help.

Essentials and concepts you'll need to know for working with WinSpector.

Tasks and procedures you need to know when using WinSpector.

Glossary Definition of terms used in WinSpector.

Shortcut: Use the **Search** button at the top of this Help window if you're ready to look for something by name.

Essentials

TOOLHELP.DLL

WinSpector uses TOOLHELP.DLL, a Microsoft Windows debugging utility. TOOLHELP.DLL must be from Windows version 3.1 or later and be included in your path.

DFA Utility

The DFA utility post processes Turbo Debugger information gathered by WinSpector at the time of the exception. If report information is set to Post-mortem Dump, WinSpector writes a WINSPCTR.BIN file at the time of the exception. DFA can then be used to translate the WINSPCTR.BIN file into a useful format.

DFA Output

The DFA utility writes a file only if Turbo Debugger information exists for the file in the stack frame. The DFA output file (DFA.OUT) has a stack trace similar to the one in the WinSpector log file, except that it has:

- function names
- line numbers
- local and global variables
- data segments and their values (including the stack segment).

After a WINSPCTR.BIN file has been used, you may want to rename or delete the DFA.OUT and WINSPCTR.LOG files. A WINSPCTR.BIN file is written for each Windows session. Renaming DFA.OUT and WINSPCTR.LOG allows WinSpector to handle more than one exception in a session.

Used with WINSPCTR.LOG

When used with the WINSPCTR.LOG file alone, DFA gives minimal stack trace information, such as addresses. Source filenames and line numbers are added to the report when Turbo Debugger information (a .tds file) is present either in the executable file or in a separate file.

Used with WINSPCTR.BIN

When used with the WINSPCTR.BIN file, DFA makes additional information available.

- Stack based variables are added to the log, including structures and arrays.
- Variable types, values, and addresses are listed by function.

If a Turbo Debugger .tds file is present, for each stack frame, DFA reports:

Section one

- source file
- line number
- local variables
- parameters

Section two

- module name for the task with the fault
- filenames
- logical segments
- their selectors
- whether it's data or code

Section three

- global variables
- static variables
- their values at time of the exception

Syntax

DFA [option] <WINSPCTR.LOG> [WINSPCTR.BIN]

The WINSPCTR.LOG file is required. With WINSPCTR.LOG, you get source file and line numbers. With WINSPCTR.BIN (optional), you get variable information.

Command line options for DFA:

/0[outputfile] (DFA.OUT default)
/D (forces DFA to write out a hex dump of the saved data segments)

Since DFA is used outside of WinSpector, to print this topic for future reference, choose Print Topic from the Help File menu.

See Also

BUILDSYM Utility
EXEMAP Utility
Interpreting the log file
Post-mortem Dump
TMAPSYM Utility

Tasks

Creating .SYM files - an overview
Creating .SYM files with the BUILDSYM utility
Creating .SYM files from existing .MAP files
Creating .MAP files for Windows format files
Creating Turbo Debugger Information
Interpreting the log file
Optimizing the report
Post processing Turbo Debugger information with the DFA Utility
Setting Preferences
Understanding addresses and selectors
Using WinSpector

Setting Preferences

WinSpector's options can be set either in the Preferences dialog box or by entering commands directly into the WINSPCTR.INI file.

Directory
Viewer
Append New Reports/Overwrite Previous Report
System Information
Summary to AUX
Post-mortem Dump
Stack Frame Data
User comments

See Also

WINSPCTR.INI file - Setting Preferences
Interpreting the log file
Optimizing the report

Directory

The Directory option in the Preferences dialog box lets you decide where the log file is written. If you do not specify a directory, it defaults to the Windows directory.

To specify a directory:

1. Open the Preferences Dialog Box.
2. Enter the directory name in the Directory text box.
3. Click the OK button.

You can also add LogDir=[directory] to the WINSPCTR.INI file.

See Also

WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

Viewer

The Viewer option in the Preferences dialog box is where you specify what program to use for viewing the log file. If you do not specify a viewing program, it defaults to Window's Notepad.

If an exception has occurred during the current Windows session, click View Log on the Latest UAE dialog box or View Log File from the System Menu to see the log file. View Log runs the selected viewing program and passes the WINSPCTR.LOG file as a command line argument.

To specify a viewer:

1. Open the Preferences Dialog Box.
2. Enter the viewer in the Viewer text box.
3. Click the OK button.

You can also add LogViewer=[viewer name] to the WINSPCTR.INI file.

See Also

WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

Append New Reports/Overwrite Previous Report

The Append New Reports and Overwrite Previous Reports options in the Preferences dialog box let you either append reports to the previous log file or overwrite the previous log file when a new report is generated. The default setting is to overwrite the previous log file.

If you choose to overwrite the previous log file, the first time an exception occurs, the previous log file is overwritten. Subsequent exceptions that occur during the same Windows session will be appended.

To append reports to the previous log file:

1 Open the Preferences Dialog Box.
2 Set Log File to Append New Reports.
3 Click the OK button.

You can also add CreateNewLog=0 to the WINSPCTR.INI file.

To overwrite the previous log file:

1 Open the Preferences Dialog Box.
2 Set Log File to Overwrite Previous Reports.
3 Click the OK button.

You can also add CreateNewLog=1 to the WINSPCTR.INI file.

See Also

WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

System Information

The System Information option in the Preferences dialog box lets you add the Task List, the Module List, and information about the USER and GDI heaps to the log file. The default is to include system information in the report.

To add system information to the log file:

1 Open the Preferences Dialog Box.
2 Under Report Information, check the System Information box.
3 Click the OK button.

You can also add ShowSystemInfo=1 to the WINSPCTR.INI file.

To omit system information from the log file:

1 Open the Preferences Dialog Box.
2 Under Report Information, uncheck the System Information box.
3 Click the OK button.

You can also add ShowSystemInfo=0 to the WINSPCTR.INI file.

See Also

WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

Summary to AUX

The Summary to AUX option in the Preferences dialog box lets WinSpector write an abbreviated form of the report to the standard DOS file AUX (called STDAUX in the C library), in addition to writing the complete log file. To use this option, you need a device driver that redirects AUX to a second monitor or a terminal connected to AUX. The default is no output to AUX.

To write Summary to AUX:

1 Open the Preferences Dialog Box.
2 Under Report Information, check the Summary to AUX box.
3 Click the OK button.

You can also add LogToStdAux=1 to the WINSPCTR.INI file.

To not write Summary to AUX:

1 Open the Preferences Dialog Box.
2 Under Report Information, uncheck the Summary to AUX box.
3 Click the OK button.

You can also add LogToStdAux=0 to the WINSPCTR.INI file.

See Also

WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

Post-mortem Dump

The Post-mortem Dump option in the Preferences dialog box generates a WINSPCTR.BIN file.

The DFA utility takes a WINSPCTR.BIN file and Turbo Debugger information (.tds files). and translates the raw binary data into a useful form. It generates a file that contains stack trace similar to the one in the log file, but with function names and line numbers, as well as local and global variables.

To generate a WINSPCTR.BIN file:

1. Open the Preferences Dialog Box.
2. Under Report Information check Post-mortem Dump.
3. Click the OK button.

You can also add Post-mortemDump=1 to the WINSPCTR.INI file.

To inhibit the generation of a WINSPCTR.BIN file:

1. Open the Preferences Dialog Box.
2. Under Report Information, uncheck Post-mortem Dump.
3. Click the OK button.

You can also add Post-mortemDump=0 to the WINSPCTR.INI file.

See Also

DFA Utility
WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

Stack Frame Data

The Stack Frame Data option in the Preferences dialog box lets you add a verbose stack trace display to the log file. For each stack frame that doesn't exceed 256 bytes, a hex dump is performed, starting at the SS:BP for that frame. If there are > 256 bytes between 2 successive stack frames, the memory display is omitted for that frame. This data can be used to get the values of parameters that were passed to the function. The default is to not generate a verbose stack trace.

It is usually easier to let the DFA utility do the hard work of figuring out what your parameters are. However, for those cases where you do not have Turbo Debugger information available, a verbose trace may be helpful.

To add Stack Frame Data to the log file:

1. Open the Preferences Dialog Box.
2. Under Report Information, check the Stack Frame Data box.
3. Click the OK button.

You can also add ShowStackInfo=1 to the WINSPCTR.INI file.

To omit Stack Frame Data from the log file:

- Open the Preferences Dialog Box.
- Under Report Information, uncheck the Stack Frame Data box.
- Click the OK button.

You can also add ShowStackInfo=0 to the WINSPCTR.INI file.

See Also

DFA Utility
WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

User Comments

The User Comments option in the Preferences dialog box lets you enter information about what was happening at the time of the exception. A dialog box is displayed immediately after the exception log is written and comments about what was happening can be entered at that time. Your comments are then appended to the log file.

To add user comments to the log file:

1 Open the Preferences Dialog Box.
2 Under Report Information, check the User Comments box.
3 Click the OK button.

You can also add ShowUserInfo=1 to the WINSPCTR.INI file.

To omit user comments from the log file:

1 Open the Preferences Dialog Box.
2 Under Report Information, uncheck the User comments data box.
3 Click the OK button.

You can also add ShowUserInfo=0 to the WINSPCTR.INI file.

See Also

WINSPCTR.INI file - Setting Preferences
Help Contents
Preferences - Setting

Optimizing the Report

The availability of .SYM files and Turbo Debugger information greatly enhances the WinSpector error report log file. There are utilities available to create and use both kinds of symbolic information. Click the underlined text to find out more about creating and using .SYM files and Turbo Debugger information.

.SYM Files - Creating
Turbo Debugger Information

See Also
BUILDSYM Utility
DFA Utility
EXEMAP Utility
Interpreting the log file
TMAPSYM Utility

BUILDSYM Utility

What BUILDSYM is:
The BUILDSYM utility offers a convenient way to create .SYM files for one or several programs in a directory.

When .SYM files are not available, creating them without BUILDSYM is a two step process:

1 Use the EXEMAP utility on the program to make a .MAP file
2 Use the TMAPSYM utility on the .MAP file to make a .SYM file

BUILDSYM uses EXEMAP and TMAPSYM, but you enter only one command to complete the process. BUILDSYM also erases .MAP files from your directory after .SYM files are created. BUILDSYM's support for wild cards in the syntax lets you create .SYM files for part or all of a directory by entering a single command.

BUILDSYM requires that EXEMAP and TMAPSYM utilities be in your search path. Resulting .SYM files are placed in the current directory. In order for WinSpector to find a .SYM file, it must be in the same directory as the Windows program.

What BUILDSYM does

- BUILDSYM verifies that the target files are really windows files (if they're not, BUILDSYM leaves them alone)
- BUILDSYM calls EXEMAP to create .MAP files
- BUILDSYM verifies that .MAP files were created
- BUILDSYM calls TMAPSYM, passing the names of the new .MAP files
- TMAPSYM creates .SYM files
- BUILDSYM deletes the .MAP files (which are no longer needed)

Syntax

```
BUILDSYM    filename
```

(DOS wild cards are supported in the filename portion of the syntax.)

Tips
BUILDSYM is particularly useful if you don't have the Microsoft Software Development Kit (SDK) or even if you have SDK, but want additional information like references to otherwise undocumented functions.

Borland precompiled header files use a .SYM extent and could be inadvertently overwritten when generating a .SYM file. If you are using the command line compiler, there is an option to rename the header file so that there is no naming conflict.

BUILDSYM overwrites any existing .SYM file. To be safe, copy existing .SYM files before using BUILDSYM or TMAPSYM.

Since BUILDSYM is used outside of WinSpector, to print this topic for future reference, choose Print Topic from the Help File menu.

See Also
DFA Utility
EXEMAP Utility
Interpreting the log file
TMAPSYM Utility

TMAPSYM Utility

What it is:
TMAPSYM creates .SYM files from existing .MAP files (created either by TLINK or by the EXEMAP utility). The resulting .SYM files make public functions, variable names, and functions in the entry table of the executable available to WinSpector. Constants and line number information is not included in a TMAPSYM generated .SYM file.

Syntax
```
TMAPSYM     filename.[MAP]
```

The .MAP extention is optional.

Tips:
When .MAP files are not available, the BUILDSYM utility lets you use a single command to create .SYM files for programs. BUILDSYM uses both EXEMAP (to make the .MAP files) and TMAPSYM (to make the .SYM files). BUILDSYM works on one, several, or all files in a directory.

Used in harmony with each other, the three WinSpector utilities (EXEMAP, TMAPSYM, and BUILDSYM) are particularly useful if you don't have the Microsoft Software Development Kit (SDK) or even if you have the SDK, but want additional information like references to otherwise undocumented functions.

Borland precompiled header files use a .SYM extention and could be inadvertently overwritten when generating a .SYM file. If you are using the command line compiler, there is an option to rename the header file so that there is no naming conflict.

BUILDSYM overwrites any existing .SYM file. To be safe, copy existing .SYM files before using BUILDSYM or TMAPSYM.

Since TMAPSYM is used outside of WinSpector, to print this topic for future reference, choose Print Topic from the Help File menu.

See Also
BUILDSYM Utility
DFA Utility
EXEMAP Utility
Interpreting the log file

EXEMAP Utility

What it is:
EXEMAP creates .MAP files for Windows format files. A .MAP file can be used to create a .SYM file, which can then be used by WinSpector to enhance error reporting. This can be especially useful when using window .DLL files or other programs for which the source or a .MAP file is not readily available.

Although the resulting .MAP file can not be as complete as one generated by the linker, it does include addresses for exported public functions.

Syntax
There is a command line option that lets you specify an output file name. If no output file name is given, exefile.MAP is the default.

```
EXEMAP      <exefilename> [output mapfile]
```

If [output mapfile] is not given, it defaults to "exefile.map".

Tips
The BUILDSYM utility lets you use a single command to create .SYM files for programs. BUILDSYM uses both EXEMAP (to make the .MAP files) and TMAPSYM (to make the .SYM files). BUILDSYM works on one, several, or all files in a directory.

Used in harmony with each other, the three WinSpector utilities (EXEMAP, TMAPSYM, and BUILDSYM) are particularly useful when you don't have the Microsoft Software Development Kit (SDK) or if you have the SDK, but want additional information like references to otherwise undocumented functions.

Since EXEMAP is used outside of WinSpector, to print this topic for future reference, choose Print Topic from the Help File menu.

See Also
BUILDSYM Utility
DFA Utility
Interpreting the log file
TMAPSYM Utility

Interpreting the Log File

To get help on an item, click the underlined text.

Addresses and Selectors
First line of log
Second line of log
Disassembly Section
Stack Trace Section
Register Section
Message Queue
Tasks Section
Modules Section
USER and GDI Heap Information
System Information Section

See Also

Exception Types
Optimizing the Report

Addresses and Selectors

The concept of address and selectors is important in understanding the log file. Windows .EXE files are referred to as new EXE files, because they have a different format than traditional MS-DOS files. In the log file, addresses are given in terms of logical addresses and physical addresses (or selectors). The logical address appears first, followed by the physical address in ( )'s.

See Also

Interpreting the log file
Optimizing the Report

First Line of Log

The first line in the log file gives the date and time that the exception occurred, as well as the title of the exception report.

See Also
Interpreting the log file

Second Line of Log

The second line lists

- what type of exception occurred
- the module name
- the logical address
- the physical address
- the current task at the time of the exception.

If the stack pointer is too small at the time of the exception, TOOLHELP. DLL automatically switches the stack. When this happens, the message "Stack Switched" is appended to the end of the second line of the log.

See Also

Exception Types
Interpreting the log file

Stack Trace Section

The first line of the Stack Trace section of the log identifies the function or procedure that was executing at the time of the exception. Stack Trace information includes

- frame number
- module name
- the name of the closest function before the address of the one that caused the exception, plus a number indicating how far away you were from that function (this information is present only if a .SYM file is present)
- logical and physical address for the stack frame
- where your program comes back to after the call.

When WinSpector gives function names, it looks in the .SYM file for the closest symbol name that appears before the address in the call stack. Some .SYM files do not contain information for all functions. Thus, the function name appearing in the log file will be that of the closest function in the .SYM file with an address preceding the frame address. If the offset field appears to be too high, function names are suspect.

See Also

Interpreting the log file
Optimizing the Report
.SYM Files - Creating

Disassembly Section

The first line of the disassembly section in the log file identifies the assembly language instruction that caused the exception.

This is followed by the next few instructions in the program. These subsequent commands are listed to provide a point of reference for finding the task which caused the exception.

See Also
Interpreting the log file

Register Section

The register section of the log file gives the values that are in the standard registers at the time of the exception. Limits and access rights are given for the CS, DS, ES, and SS registers.

See Also
Interpreting the log file

Message Queue Section

The message queue section of the log file gives the last message actually received in the middle of processing. Also given is a list of any messages that were waiting in the queue at the time of the exception.

Listed is the following information

- window handle (identifies what window)
- message ID number (identifies what it was)
- two parameters (present for any given window).

What is recorded in the message queue section may not really be the last message the program received. Windows may bypass the message queue, i.e., SendMessage(). Keep that in mind when using message queue information.

See Also
Interpreting the log file

Tasks Section

The Tasks section of the log file lists all programs running in the system at the time of the exception.

Given is

- complete path for executor file
- module name
- windows module handle
- task handle
- what the data segment value was for the task (the instance handle).

See Also

Interpreting the log file

Modules Section

The Modules section of the log lists the modules that were running at the time of the exception.

Given is

- path for executor file
- the date
- the file size
- module name
- module handle
- reference count (how many times the module is in use).

See Also
Interpreting the log file

USER and GDI Heap Information

The USER and GDI heap information section of the log shows what percentage of both the USER and GDI heap was available at the time of the exception.

See Also
Interpreting the log file

System Information Section

The System Information section of the log file shows the mode and windows version under which your program was run.

Also given is

- CPU information
- largest free memory block
- total linear memory space
- free linear memory space
- swap file pages.

See Also
Interpreting the log file

Exception Types

Line two of the log file gives the number of the CPU exception that just occurred. Exception type numbers are described in detail in the INTEL documentation.

Frequently encountered exception types:

0 Happens during a DIV or an IDIV interaction when the divisor is 0.

12 Usually happens when there is too little room on the stack to proceed.

13 All protection errors which don't cause another exception cause an exception 13. This includes (but is not limited to):

- exceeding segment limit (like "array out of bounds") in DS, ES, or other segments.
- transferring execution to a non-executable segment (bad function pointer).
- accessing DS, ES, FS, or GS registers containing a null selector.
- 0 in the segment register of the log file.

See Also

Interpreting the log file

The Latest UAE Logged Dialog Box

WinSpector intercepts unrecoverable application errors (UAE's) or general protection errors and writes an error report log file. This log file can assist you in finding the cause of the UAE. If an exception has occurred, the date and time of the exception is displayed in the Latest UAE Logged Dialog Box.

To view the log file, click View log.

To change log file or report preferences, choose Set Pref.

- Log preferences include where it is written, what editor is used to view the log, and whether or not to overwrite the existing log when an exception occurs.
- Report preferences include adding system information to the log, writing to AUX, generate a binary file for post processing, adding stack frame data to the log, and adding an option to enter user comments to the log at the time of the exception.

See Also

Help Contents
Interpreting the log file
Optimizing the report
Preferences - Setting

The Preferences Dialog Box

The WinSpector Preferences dialog box lets you set both log file preferences and report information preferences.

Log preferences include where it's written, what editor is used to view it, and whether or not WinSpector overwrites or appends to the existing log when an exception occurs.

Report preferences include adding system information to the log, writing to AUX, generating a binary file for post processing, adding stack frame data to the log, and adding the ability to enter user comments right when the exception occurs.

Directory
Enter the name of the directory where the log file will be written.

Viewer
Enter the name of the editor to use when viewing the log file.

Append New Reports
Set Log File to Append New Reports to append the latest UAE report to the end of the existing log file.

Overwrite Previous Report
Set Log File to Overwrite Previous Report to overwrite the existing log file with the latest UAE report.

System Information
Set Report Information to System information to add system information to the log.

Summary to AUX
Set Report Information to Summary to AUX to write an abbreviated form of the report to AUX.

Post-mortem Dump
Set Report Information to Post-mortem Dump to generate a WINSPCTR.BIN file for post-mortem processing.

Stack Frame Data
Set Report Information to Stack Trace to add a verbose stack trace display to the log file.

User Comments
Set Report information to User Comments so you can enter user comments right when the exception occurred and have them added to the log file.

See Also
WINSPCTR.INI file - Setting Preferences
Help Contents
Interpreting the log file
Optimizing the Report
Preferences - Setting

WINSPCTR.LOG

WinSpector creates a report file (winspctr.log) when an unrecoverable application error (UAE) or general protection error occurs. The log file can be helpful in finding out what caused the exception. The latest log file can be viewed directly from the Latest UAE dialog box. To specify a viewer, click Set Prefs. and enter the viewer of your choice.

Glossary

BUILDSYM
call stack
DFA utility
Directory
WINSPCTR.BIN
WINSPCTR.INI
WINSPCTR.LOG
EXEMAP utility
free linear memory space
largest free memory block
logical addresses
physical addresses
Post-mortem Dump
selectors
stack information
swap file pages
.SYM files
system information
TOOLHELP.DLL
TMAPSYM utility
total linear memory space
user comments
Viewer

Introduction

WinSpector
WinSpector and its utilities help you perform a post-mortem examination of your Microsoft Windows programming Unrecoverable Application Errors (UAE) or General Protection Errors.

How to use it
1. Run WinSpector.
2. When an exception occurs, WinSpector writes a log file to your disk.
3. A Microsoft Windows "Unrecoverable Application Error" box or "General Protection Error" box is displayed.
4. Click the OK or Close button.
5. A WinSpector dialog box with a brief exception report is displayed.
6. Click the OK button.
7. Read the log file. It contains information that can help you find the cause of the exception.

What it can show you
- the call stack.
- function and procedures names in the call stack (with a little help from you).
- CPU registers.
- a disassembly of the instructions.
- Windows information.

Getting started
Before using WinSpector, be sure that TOOLHELP.DLL (Windows 3.1 or later) is in your search path. To be safe, don't have other exception debugging tools running concurrent with WinSpector (except TDW).

The easiest way to use WinSpector is to put it in the "load=" section of your WIN.INI file. Upon starting, WinSpector will minimize. No additional interaction is required.

Getting the most out of WinSpector
WinSpector can be configured to suit your needs.

Four options allow you to gather specific information.
- Set System Information
- Stack Frame Data
- User Comments
- Post-mortem Dump

Three pre-processing utilities help you make .SYM files available prior to the exception. WinSpector uses the .SYM files to greatly enhance the UAE report.
- BUILDSYM Utility
- TMAPSYM Utility
- EXEMAP Utility

The DFA Utility, WinSpector's post-processing utility, can help you utilize Turbo Debugger symbolic information to further enhance the readability of available UAE information.

See Also
BUILDSYM Utility
DFA Utility
EXEMAP Utility
Optimizing the Report
Preferences - Setting
.SYM Files - Creating
TMAPSYM Utility

System Information

The System Information option lets you add the Task List, the Module List, and information about the USER and GDI heaps to the log file. The default is to include system information in the report.

Stack Frame Data

The Stack Frame Data option adds a verbose stack trace display to the end of the log file. Each entry displays the memory at a positive offset from the SS:BP for that stack frame. If there are > 256 bytes between 2 successive stack frames, the memory display is omitted for that frame. The default is to not generate a verbose stack trace.

User Comments

The User Comments option lets you enter user comments at the time of the exception about what was happening when the exception occurred. Those comments are then appended to the log.

Post-mortem Dump

The Post-mortem Dump Option generates a WINSPCTR.BIN file. This raw data file can then be translated into a useful format by the DFA utility.

BUILDSYM Utility

The BUILDSYM utility automates the process of building .SYM files for your programs. It calls the EXEMAP utility to create .MAP files, then calls the TMAPSYM utility to make .SYM files. BUILDSYM supports wild cards in the syntax and can be used to create .SYM files for a single file, a partial directory, or an entire directory.

TMAPSYM Utility

TMAPSYM creates .SYM files from existing .MAP files (created either by TLINK or by the EXEMAP utility). The resulting .SYM files make public functions, variable names, and functions in the entry table of the executable available to WinSpector. Constants and line number information is not included in the .SYM file.

EXEMAP Utility

EXEMAP creates .MAP files for new EXE file format files. When .MAP files are available for programs, .SYM files can be created. WinSpector uses .SYM files to enhance the error report. EXEMAP is particularly useful when source code for a new EXE file is not available.

DFA Utility

The DFA utility combines WINSPECTR.LOG with WINSPCTR.BIN into a useful format. WinSpector writes a WINSPCTR.BIN file if report information is set to Post-mortem Dump.

.SYM Files - Creating

The availability of .SYM files greatly enhance the WinSpector error report log file. If .SYM files are present, they are used directly by WinSpector and no additional utility is required. The .SYM file must be in the same directory as the corresponding .EXE or .DLL.

Creating .SYM files for your own programs

First create a .MAP file,

- If you're using a Borland IDE, select the Linker options that produce a MAP file.
- If you're letting BCC (Borland command-line C++ Compiler) invoke TLINK, use the "-M" option on the BCC command line.
- If you're invoking TLINK yourself, use the "/m" command line option, and make sure that you're not using "/x" (no map file).
- If you're using BPC (Borland Pascal Compiler) or TPC (Turbo Pascal Compiler), use the /GD command line switch.
- If you're using another compiler, do whatever is necessary to create a .MAP file with Public symbols.

Then, create a .SYM file from the .MAP file.

- If your .MAP file is from a Borland product, use TMAPSYM to create the .SYM file.
- If you're using another compiler, use MAPSYM, provided with the Microsoft SDK to create the .SYM file.
- If you're using a makefile, you can add TMAPSYM as another rule command after the link.

Creating .SYM files for programs when you don't have the source:

The easiest way to create a .SYM file for programs that you don't have the source code for (like USER.EXE and GDI.EXE) is to use the BUILDSYM utility. BUILDSYM uses both the EXEMAP and the TMAPSYM utilities, so make sure all three (BUILDSYM, EXEMAP, and TMAPSYM) are in your path.

You'll probably want to build .SYM files in the WINDOWS directory, as well as the SYSTEM subdirectory below it. Although the .SYM files created by BUILDSYM are usually superior to the Microsoft SDK provided .SYM files, you may want to back up the SDK provided .SYM files before overwriting them with .SYM files created by BUILDSYM.

See Also:

BUILDSYM
EXEMAP
Interpreting the log file
Optimizing the Report
TMAPSYM Utility

Turbo Debugger Information

Turbo Debugger information contains line numbers and can help DFA find static functions that might not be included in a .SYM file. In addition to function and procedure names, DFA can give source file and line numbers for call stack entries. You'll be able to see the names and values of your program's variables.

To create Turbo Debugger debug information for your programs

- If you're using a Borland IDE, make sure that the option to generate debug information is on.
- If you're letting BCC invoke TLINK, specify the "-v" option in the BCC command line.
- If you're invoking TLINK yourself, specify the /V option.
- If you're using TPCW, use the /V switch.
- If you're using a compiler that generates CodeView information, and have a copy of Turbo Debugger that has TDCONVRT included, you may be able to use TDCONVRT to convert the CodeView information to TD information.

The DFA Utility makes it possible to post-process the log and take advantage of Turbo Debugger information.

See Also:
BUILDSYM Utility
DFA Utility
EXEMAP Utility
Interpreting the log file
Optimizing the Report
TMAPSYM Utility

.SYM Files

.SYM files contain symbolic information that WinSpector can use at the time of the exception. They are created with TLINK or with the WinSpector utilities. Use the EXEMAP utility first to create a .MAP file if one does not exist. Use the TMAPSYM utility next to create a .SYM file from the .MAP file.

The BUILDSYM utility automates the building of .SYM files for one or many programs right at the directory level, by using both EXEMAP and TMAPSYM for you.

Largest Free Memory Block

The largest free block of contiguous linear memory in the system in bytes.

Total Free Memory Space

The size of total linear address space in 14K pages.

Free Linear Memory Space

The amount of free memory in the linear address space in 14 K pages.

Swap File Pages

The number of 14K pages in the system swap file.

WINSPCTR.BIN file

WINSPCTR.BIN is a raw data file which is generated by WinSpector at the time of the exception and can be processed by the DFA utility. WINSPCTR.BIN is generated when the Post-mortem Dump option is set.

The Call Stack

A list of the functions that were being called when the exception occurred.

Logical Addresses

When a new EXE file is linked, each segment is placed in a different section of the file and a segment table is created allowing for rapid segment access. You can find out the number of segments, their size and other information by running TDUMP (a Turbo Debugger utility) on the file. The same information as is available in a .MAP file is generated for the .EXE file.

A segment's position in the Windows segment table is a logical address. Logical addresses are used in the Public section of a .MAP file. For any given .EXE or .DLL, the logical address will not change.

A logical address is comprised of

- module name
- logical segment
- offset.

Physical Addresses (Selectors)

When a program is loaded, Windows allocates space for each logical segment and assigns it a unique selector.

Both a selector and an offset combined make up a physical address for a logical segment. A typical physical address might be 09CD:65EA. Physical addresses are what the CPU uses. The selector Windows uses for a particular logical segment can change between different invocations of the .EXE or .DLL.

WINSPCTR.INI File - Setting Preferences

WinSpector preferences can be set directly via the Preferences dialog box or you can fine tune the WinSpector reports right in the WINSPCTR.INI file. To find out more about setting each preference, click the underlined text for the desired option.

Directory
Viewer
Append New Reports/Overwrite Previous Report
System Information
Summary to AUX
Post-mortem Dump
Stack Frame Data
User comments

See Also

Interpreting the log file
Optimizing the report

WINSPCTR.INI File

The WINSPCTR.INI file contains settings for user defined preferences, such as, in what directory the log is written, what viewer is used to see the log, and what specific information is gathered at the time of the exception.

Directory

The Directory option in the Preferences dialog box lets you decide where the log file is written.

Viewer

The Viewer option in the Preferences dialog box is where you specify what program to use for reading and viewing the log file.

User Comments

The User Comments option in the Preferences dialog box lets you enter information about what was happening at the time of the exception.

WINSPCTR.LOG

WinSpector creates a report file (winspctr.log) when an unrecoverable application error (UAE) or general protection error takes place. The log file can be helpful in finding out what caused the exception. The latest log file can be viewed directly from the Latest UAE dialog box. To specify a viewer, click Set Prefs. and enter the viewer of your choice.

TOOLHELP.DLL

WinSpector uses TOOLHELP.DLL, a Microsoft Windows debugging utility. TOOLHELP.DLL must be from Windows version 3.1 or later and be included in your path.