OutputDebugString Checker

by

Software Verify

Software Verify

Troubleshoot your software

OutputDebugString Checker

OutputDebugString usage detector for Windows Operating Systems

by Software Verify Limited

Welcome to the OutputDebugString Checker software tool. OutputDebugString Checker is software tool that scans source code looking for specific uses of OutputDebugString(). Once you have found these specific uses you can decide if they are correct for the context in which they are found.

We hope you will find this document useful.

OutputDebugString Checker Help

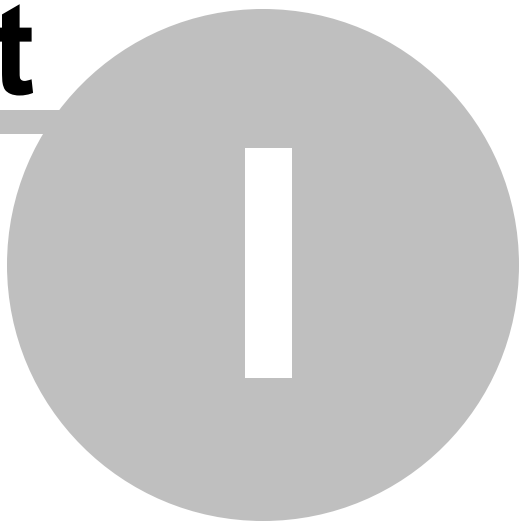
Printed: February 2025 in United Kingdom.

Table of Contents

Foreword

This is just another title page
placed between table of contents
and topics

Part



1 How to get OutputDebugString Checker

OutputDebugString Checker is free for commercial use. OutputDebugString Checker can be downloaded for Software Verify's website at https://www.softwareverify.com/product/outputdebugstring-checker/.

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

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

Contact

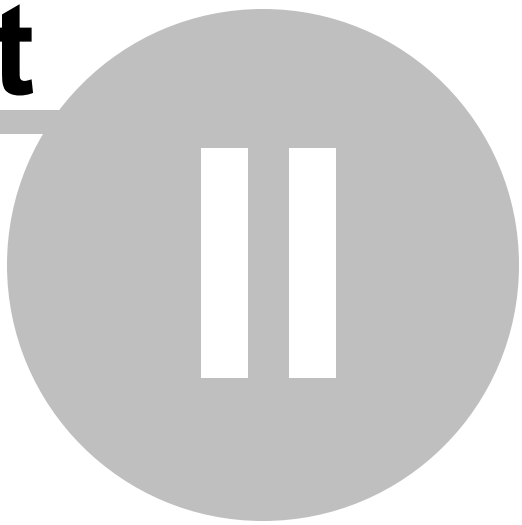
Contact Software Verify at:

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

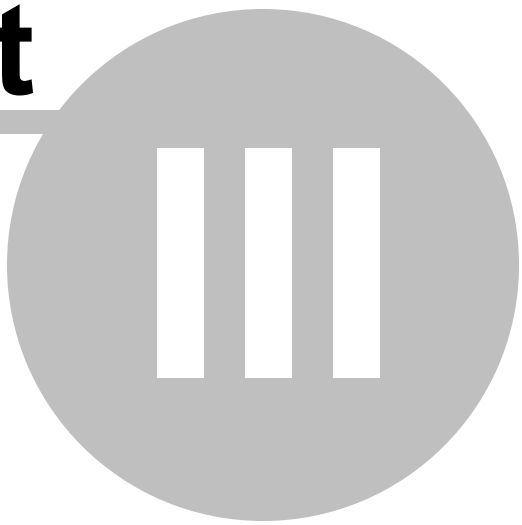


2 What does OutputDebugString Checker do?

OutputDebugString Checker scans your source code looking for calls to OutputDebugString() and reports each call with a list of any conditional compilation directives. This allows you to determine if a call to OutputDebugString() has been left in the code by accident (and should thus be removed) or should be present in the code.

As well as checking for OutputDebugString() calls, you can also check for any functions you choose to specify.

Part



3 Default Conditionals

The default settings will cause OutputDebugString Checker to recognise the following conditional compilation statements:

_DEBUG

What if I've got my own conditionals I need to check for?

If you've got your own conditionals you can specify these using the settings dialog or the command line.

Part

4 Default Functions

The default settings will cause OutputDebugString Checker to search for the following functions:

Win32

OutputDebugString
OutputDebugStringA
OutputDebugStringW

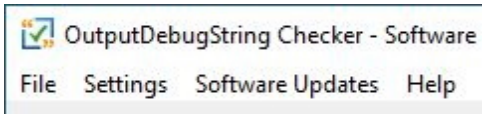
What if I've got my own functions I need to check for?

If you've got your own functions you can specify these using the settings dialog or the command line.

Part V

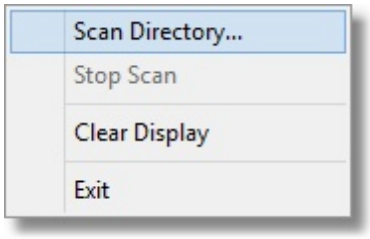
5 Menu

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

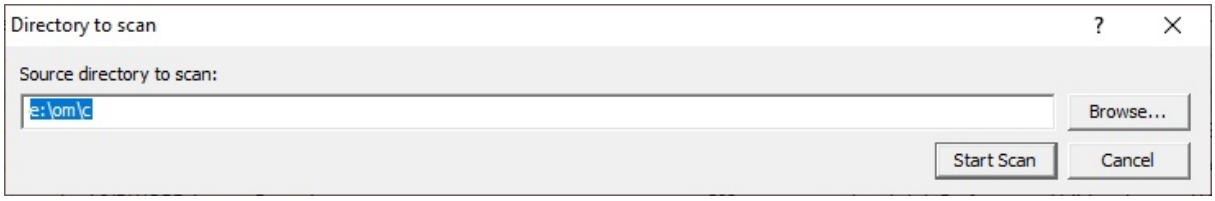


5.1 File

The File menu controls starting and stopping scans for calls to OutputDebugString, clearing the display and exiting the program.



File menu ➔ **Scan Directory...** ➔ displays a directory chooser then starts scanning that directory for calls to the functions listed in the settings.



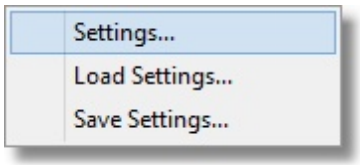
File menu ➔ **Stop scan** ➔ stops any scan that is in progress

File menu ➔ **Clear Display** ➔ clear all results from a previous scan and to remove any source code from the source code editing window.

File menu ➔ **Exit** ➔ closes OutputDebugString Checker

5.2 Edit

The Edit menu controls editing settings, loading settings and saving settings.



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

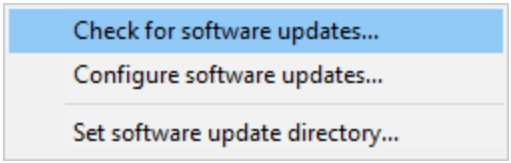
Edit menu → **Load Settings...** → loads previously saved settings.

Edit menu → **Save Settings...** → saves settings.

5.3 Software Updates

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

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

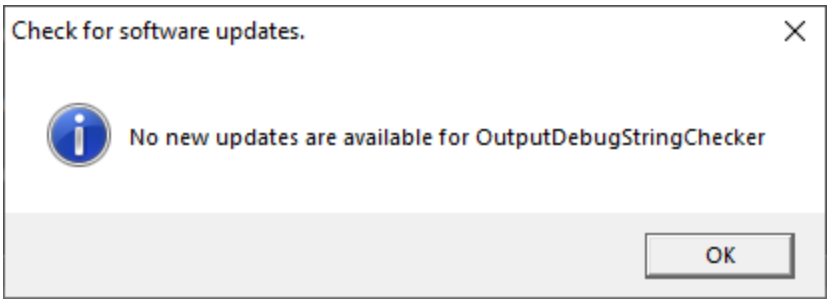


Software Updates menu → **Check for software updates** → checks for updates and shows the software update dialog if any exist

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

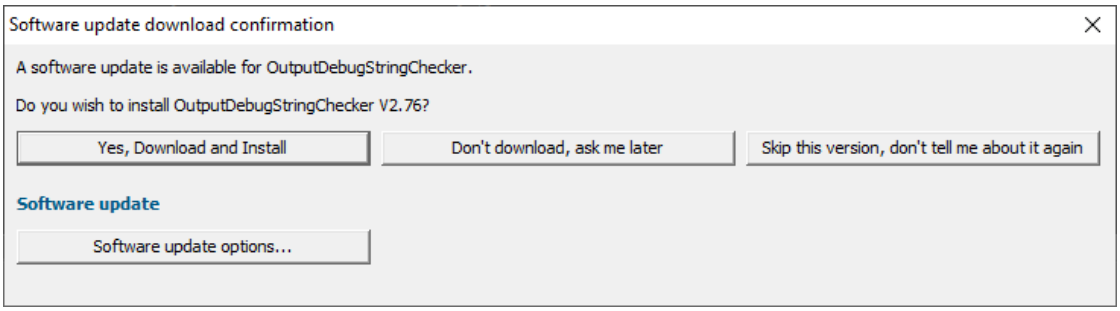
Before updating the software, close the help manual, and end any active session by closing target programs.

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

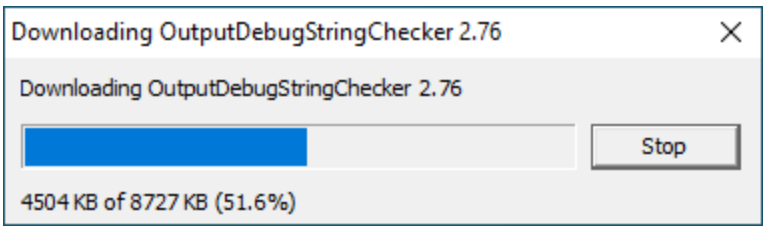


Software Update dialog

If a software update is available for OutputDebugString Checker you'll see the software update dialog.



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



  Once the update has downloaded, OutputDebugString Checker 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 OutputDebugString Checker
- **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 OutputDebugString Checker as follows:

- Ensure OutputDebugString Checker 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

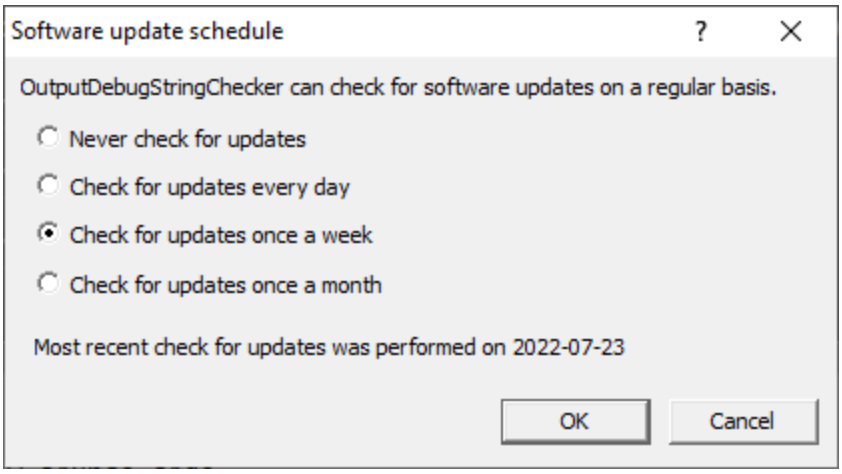
OutputDebugString Checker can automatically check to see if a new version of OutputDebugString Checker is available for downloading.

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.

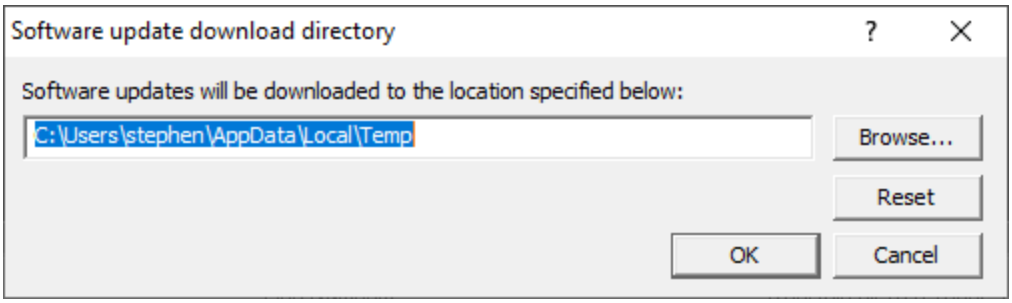


Software update directory

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)

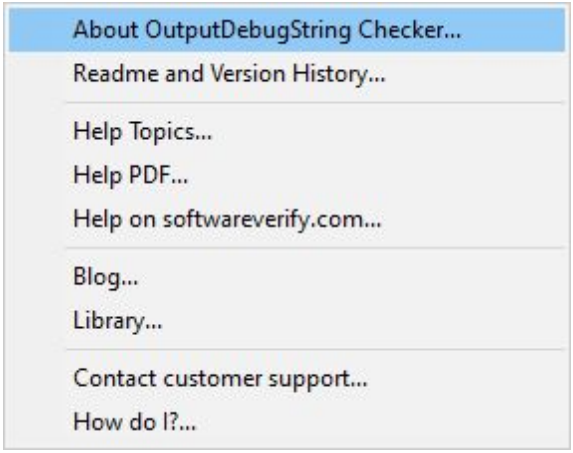
When modifying the download directory, you should ensure the directory will continue to be valid. Updates may no longer occur if the download location is later invalidated.

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

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

5.4 Help

The Help menu controls displaying this help document and displaying information about OutputDebugString Checker.



Help menu **About OutputDebugString Checker...** displays information about OutputDebugString Checker.

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

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

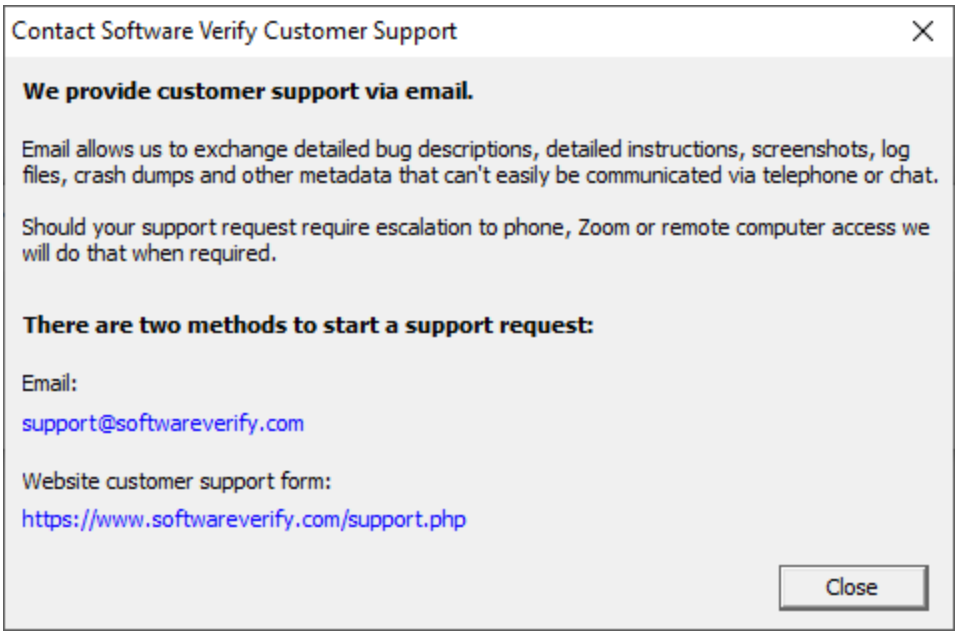
Help menu ⇨ **Help PDF...** ⇨ displays this help file in PDF format.

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.

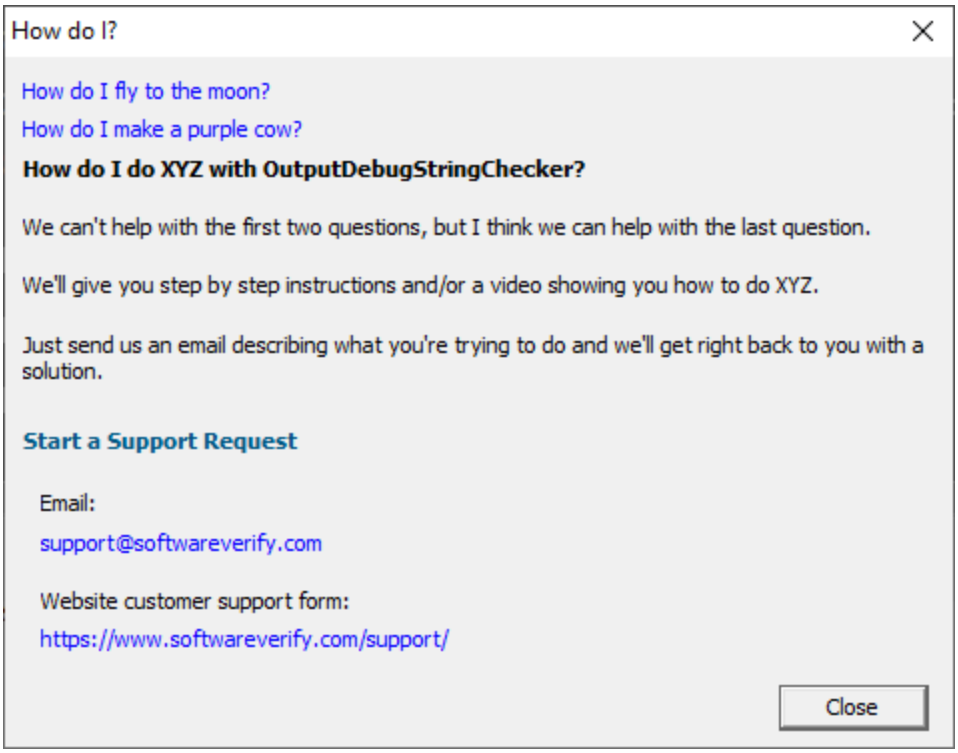
Help menu ⇨ **Library...** ⇨ display the Software Verify library - our best blog articles grouped by related topics.

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 VI

6 The user interface

The OutputDebugString Checker user interface is split into three sections:

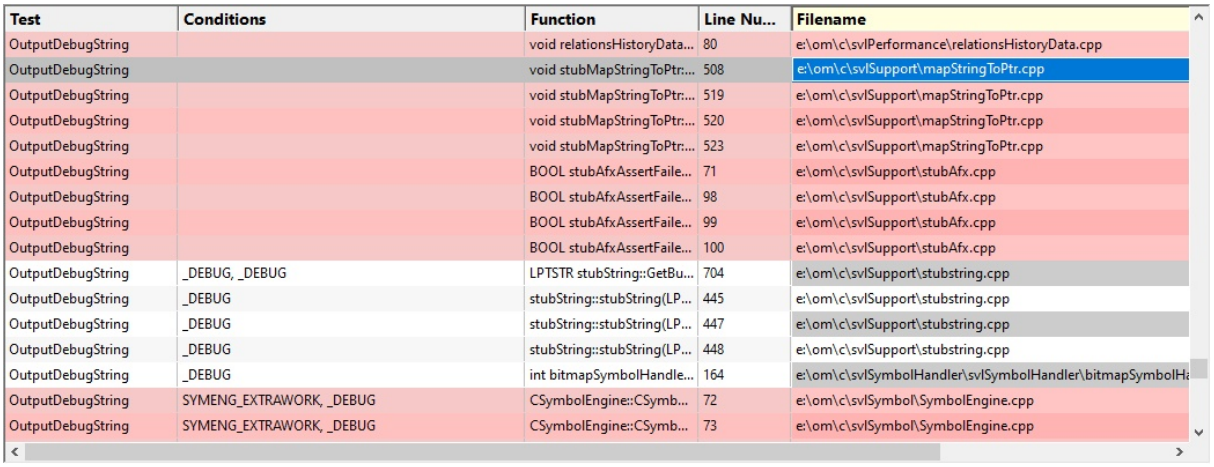
- Menu
- Error report grid
- Source code viewer

Error Report Grid

The error report grid displays one warning per line. Each line contains the test function name, any conditional compilation values in force, the name of the function containing the test function, the line number and the filename of the error.

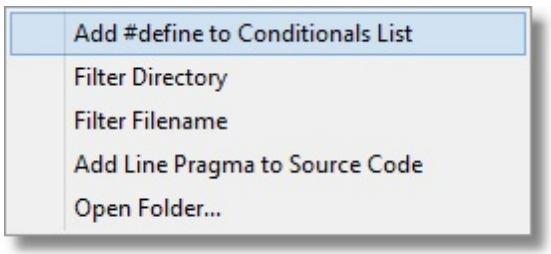
You can sort the data by clicking on any column header. Click again to reverse the sort direction.

Click on any line to display the source code in the lower window. Double click to edit the source code in Visual Studio.



Test	Conditions	Function	Line Nu...	Filename
OutputDebugString		void relationsHistoryData...	80	e:\om\c\svlPerformance\relationsHistoryData.cpp
OutputDebugString		void stubMapStringToPtr:...	508	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		void stubMapStringToPtr:...	519	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		void stubMapStringToPtr:...	520	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		void stubMapStringToPtr:...	523	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	71	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	98	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	99	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	100	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString	_DEBUG, _DEBUG	LPTSTR stubString::GetBu...	704	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	stubString::stubString(LP...	445	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	stubString::stubString(LP...	447	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	stubString::stubString(LP...	448	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	int bitmapSymbolHandle...	164	e:\om\c\svlSymbolHandler\svlSymbolHandler\bitmapSymbolHa
OutputDebugString	SYMENG_EXTRAWORK, _DEBUG	CSymbolEngine::CSymb...	72	e:\om\c\svlSymbol\SymbolEngine.cpp
OutputDebugString	SYMENG_EXTRAWORK, _DEBUG	CSymbolEngine::CSymb...	73	e:\om\c\svlSymbol\SymbolEngine.cpp

A context menu is available on the error report grid.



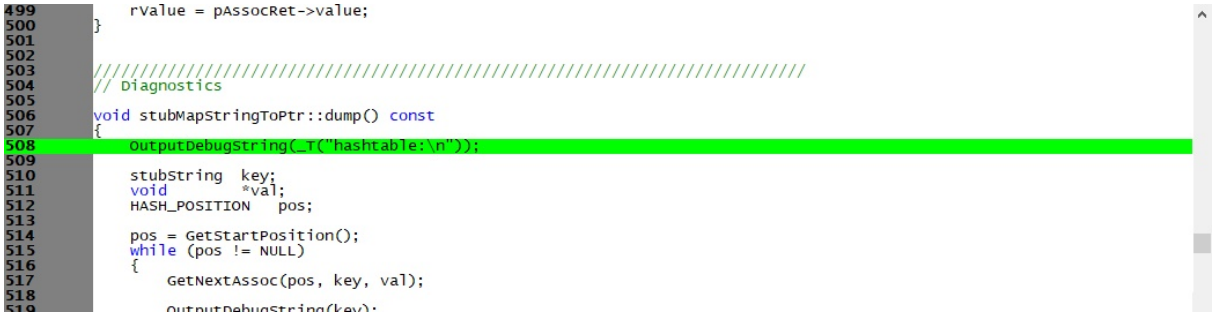
The context menu has five options:

- An option to add the conditional compilation #define value to the list of known conditional compilation values. Any values set can be edited on the Conditionals settings.

- Two options for filtering by directory and by filename. Any values set by these filters can be edited on the File Filters settings.
- An option to insert a line pragma at the end of the selected line. Any values set can be edited on the Line Pragma settings.
- An option to open an explorer window at the directory for the selected line.

Source Code Viewer

The source code viewer displays the source code with the line of interested highlighted in green.
Click on any line in the error report grid to view the source code in the source code viewer.
If you would rather edit the source code using Visual Studio, double click the line in the grid.

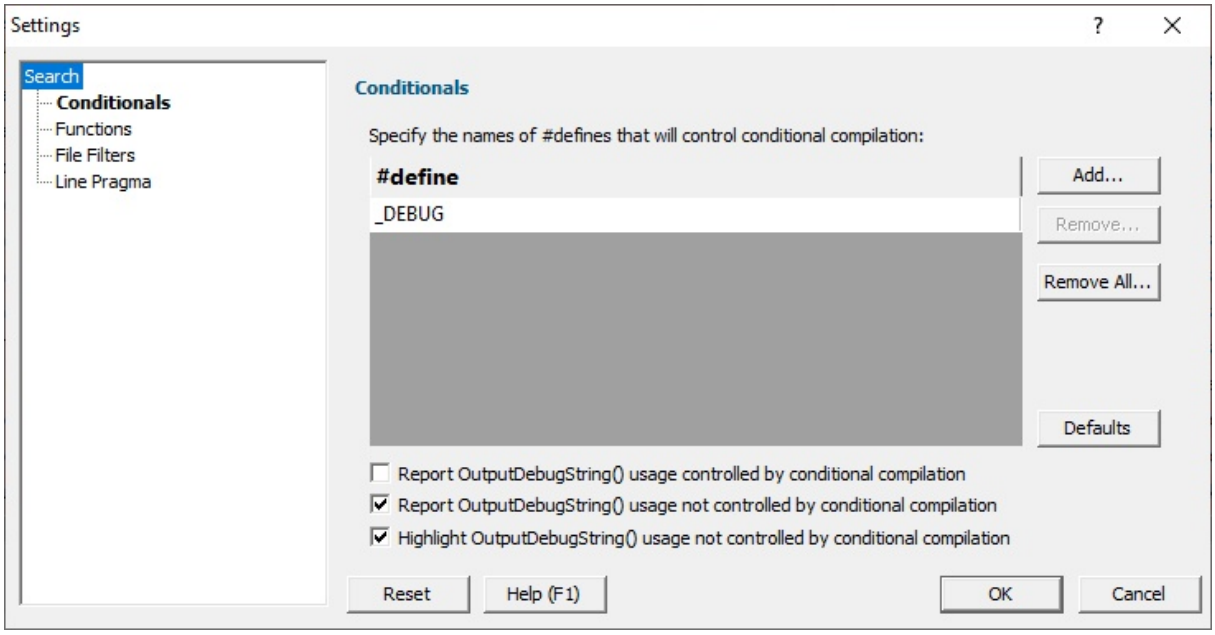


Part VII

7 Modifying the settings

The settings dialog allows you to control all OutputDebugString Checker settings.

The settings are grouped into logical choices making it easier to manage the settings.

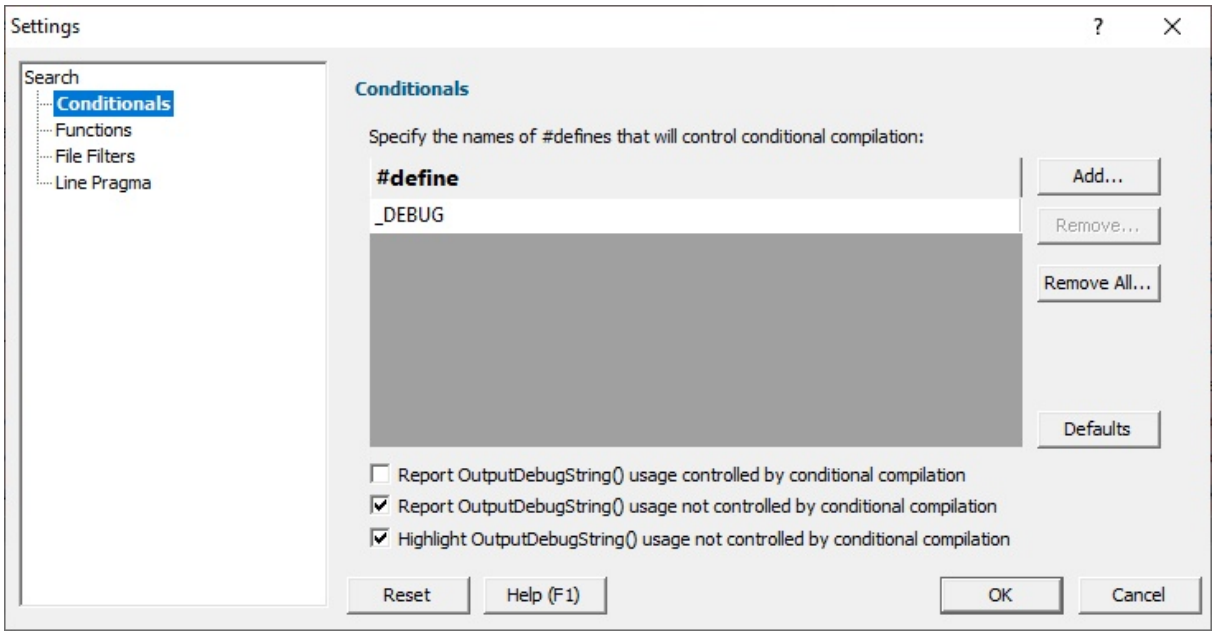


Reset

Click **Reset** to reset all settings on all pages of the settings dialog.

7.1 Conditionals

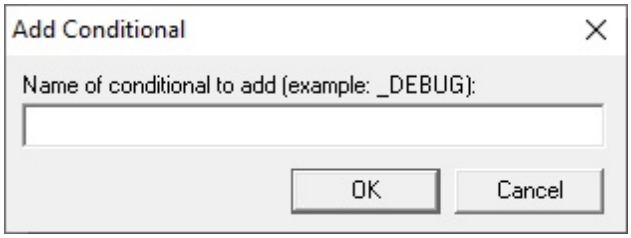
The Conditionals page of the settings dialog allows you to control how OutputDebugStringChecker responds to conditional compilation of C and C++ source code.



Conditionals

You can add and remove conditional compilation defines that OutputDebugStringChecker will recognise.

Add
Click **Add** to add a new conditional.



The Add Conditional dialog box will be displayed. Type the name of the new conditional and click OK.

Remove
Click **Remove** to remove the selected conditionals.

Remove All
Click **Remove All** to remove all conditionals.

Defaults
Click **Defaults** to restore the default conditionals.

Conditional Compilation

You can change what OutputDebugStringChecker reports by using the two **Report OutputDebugString()...** check boxes.

The first check box causes OutputDebugStringChecker to report function usage that is controlled by conditionals that are specified in the settings.

The second check box causes OutputDebugStringChecker to report function usage that is not controlled by conditionals that are specified in the settings.

You can also highlight the use of functions that are not controlled by conditionals that are specified in the settings by using the **Highlight OutputDebugString()...** check box.

Reset
Click **Reset** to reset all settings on all pages of the settings dialog.

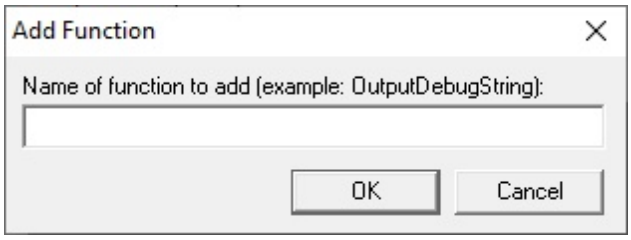
7.2 Functions

The Finctions page of the settings dialog allows you to specify functions OutputDebugString Checker will search for.

The default list of functions that OutputDebugString Checker will scan for are listed here.

You can add and remove functions that should be scanned.

Add
Click **Add** to add a new function.



The Add Function dialog box will be displayed. Type the name of the new function and click OK.

Remove
Click **Remove** to remove the selected functions.

Remove All
Click **Remove All** to remove all functions.

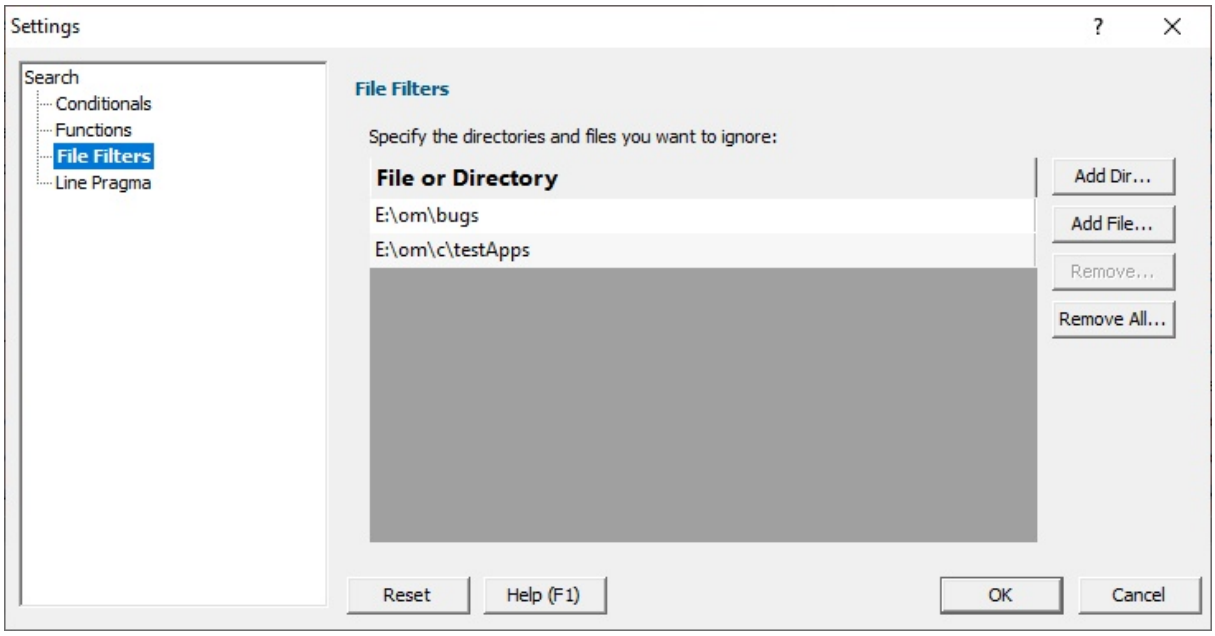
Defaults
Click **Defaults** to restore the default functions.

Reset
Click **Reset** to reset all settings on all pages of the settings dialog.

7.3 File Filters

The File Filters page of the settings dialog allows you to specify directories and files that OutputDebugString Checker will ignore when searching for functions.

This allows you to ignore directories files where you deliberately have special case usage of these functions (unit tests, development work, custom use cases) that would trigger warning results in OutputDebugString Checker that you don't want.



Add Dir...
Click **Add Dir...** to add a directory. The Microsoft directory chooser will be displayed.

Add File...
Click **Add File...** to add a file. The Microsoft file chooser will be displayed.

Remove
Click **Remove** to remove the selected directories and files.

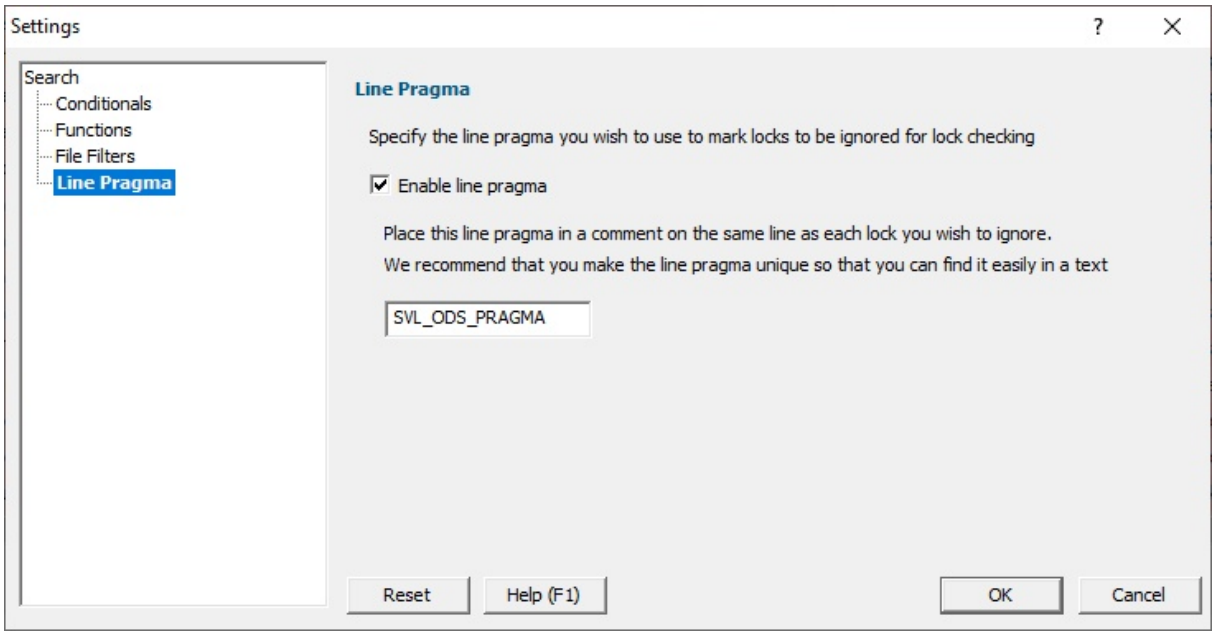
Remove All
Click **Remove All** to remove all directories and files.

Reset
Click **Reset** to reset all settings on all pages of the settings dialog.

7.4 Line Pragma

The Line Pragma page of the settings dialog allows you to specify a line pragma that can be used to mark individual locks to be ignored when checking for function call usage.

This allows you identify special use cases in your source code to avoid FALSE positives from OutputDebugString Checker.



When line pragmas are enabled any line containing the specified line pragma in a comment (single line comment or multi-line comment) will be ignored when checking for function call usage.

If the line pragma is enabled, the line pragma cannot be empty or whitespace. We recommend a value that is unique across your codebase so that you can find all line pragmas easily if you need to search for them.

Example usage of the pragma in code:

```
OutputDebugString(_T("NVIDIA Properties Dump\r\n"));      // SVL_ODS_PRAGMA
```

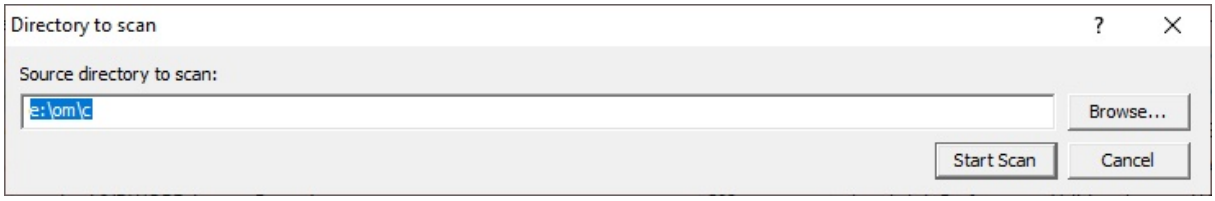
Part

8 How to use OutputDebugString Checker

To use OutputDebugString Checker follow these basic steps:

1. If you want to change the functions to scan for, **Edit > Settings...** to open the settings dialog.

2. **File > Scan directory...** to display the directory chooser.

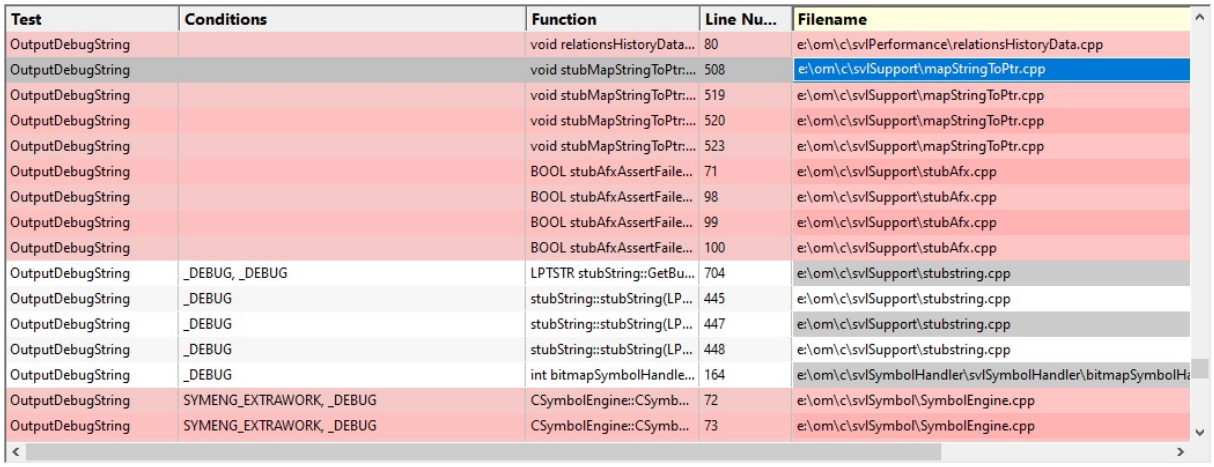


You can type the directory path or use the Microsoft directory chooser by clicking **Browse...**.

When you are happy with your choice of directory click **Start Scan** to start the process of finding the specified functions.

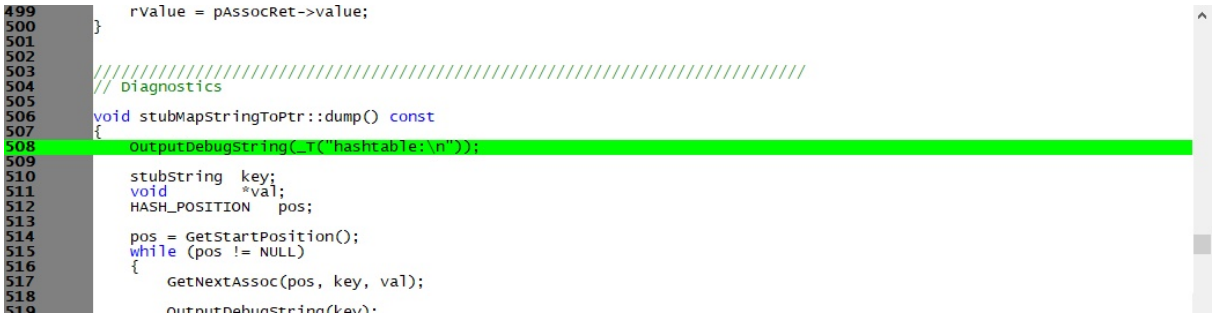
3. If at any time you want to stop the scan **File > Stop scan**.

4. Any function usages not filtered are shown in the grid.



Test	Conditions	Function	Line Nu...	Filename
OutputDebugString		void relationsHistoryData...	80	e:\om\c\svlPerformance\relationsHistoryData.cpp
OutputDebugString		void stubMapStringToPtr:...	508	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		void stubMapStringToPtr:...	519	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		void stubMapStringToPtr:...	520	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		void stubMapStringToPtr:...	523	e:\om\c\svlSupport\mapStringToPtr.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	71	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	98	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	99	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString		BOOL stubAfxAssertFaile...	100	e:\om\c\svlSupport\stubAfx.cpp
OutputDebugString	_DEBUG, _DEBUG	LPTSTR stubString::GetBu...	704	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	stubString::stubString(LP...	445	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	stubString::stubString(LP...	447	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	stubString::stubString(LP...	448	e:\om\c\svlSupport\stubstring.cpp
OutputDebugString	_DEBUG	int bitmapSymbolHandle...	164	e:\om\c\svlSymbolHandler\svlSymbolHandler\bitmapSymbolHa
OutputDebugString	SYMENG_EXTRAWORK, _DEBUG	CSymbolEngine::CSymb...	72	e:\om\c\svlSymbol\SymbolEngine.cpp
OutputDebugString	SYMENG_EXTRAWORK, _DEBUG	CSymbolEngine::CSymb...	73	e:\om\c\svlSymbol\SymbolEngine.cpp

5. Select an error shown in the grid to view the source code in the source code window.



```
499         rValue = pAssocRet->value;
500     }
501
502
503     ////////////////////////////////////////////////////////////////////////
504     // Diagnostics
505
506     void stubMapStringToPtr::dump() const
507     {
508         OutputDebugString(_T("hashtable:\n"));
509
510         stubString  key;
511         void        *val;
512         HASH_POSITION   pos;
513
514         pos = GetStartPosition();
515         while (pos != NULL)
516         {
517             GetNextAssoc(pos, key, val);
518
519             OutputDebugString(key);
```

Double click on the error to edit the source code in Visual Studio.

6. If you want to clear the results, **File > Clear**.

7. If you want to filter a location by directory, right click on the entry in the list and choose **Filter Directory**.

8. If you want to filter a location by filename, right click on the entry in the list and choose **Filter Filename**.

9. If you want to mark a function to be ignored, right click on the entry in the list and choose **Add Line Pragma to Source Code**.

10. If you want to go to the source directory, right click on the entry in the list and choose **Open Folder...**.

Part

9 Command Line Interface

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

User interface
If command line arguments are detected the user interface is not displayed.

Error reporting
When running from the command line any errors are written to the output file specified on the command line.

Return code
The process return code is 0 for no function calls found.

If there are function calls found the return code is a positive integer describing the number of function calls found.

9.1 Alphabetic Reference

/addFileFilter

Adds a filename or directory to the list of file filters OutputDebugString Checker will examine when checking for function call usage.

/addFileFilter filename
/addFileFilter directory

Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\tlcExample.cpp
Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\

/addFunction

Adds a function to the list of functions OutputDebugString Checker will check.

/addFunction functionName

Example: ./addFunction myDebugInformationFunction

/dir

Specifies the directory hierarchy that will be scanned by OutputDebugString Checker.

/dir directory

Example: /dir e:\om\c\

/linePragma

Specifies the line pragma to use if line pragmas are enabled.

/linePragma pragma

Example: /linePragma SVL_ODS_PRAGMA

Example usage of the pragma in code:

```
OutputDebugString(_T("NVIDIA Properties Dump\r\n"));            //
SVL_ODS_PRAGMA
```

/linePragmaEnable

Enables or disables the ignoring of locks if a line pragma is found in a comment on the same line as the function call.

If you use this option to enable line pragmas you must use **/linePragma** to specify a line pragma to use. If you don't specify a line pragma this option does nothing.

/linePragmaEnable:On|Off

Example: /linePragmaEnable:On
Example: /linePragmaEnable:Off

/loadSettings

Specifies a file to load that contains settings that will override the current settings for this command line search.

/loadSettings filename

Example: /loadSettings e:\tests\test1\testSettings.tlc

/output

Specifies a filename to write the results of the command line search.

If no errors are found any file with this name will be deleted.

If errors are found a text file will be written containing error information. Error information will be listed one error per line.

/output filename

Example: /output e:\tests\test1\testResults.txt

/removeAllConditionals

Removes all conditionals from the list of conditionals OutputDebugString Checker will use to check when checking for function call usage.

This option should be used before any use of /addConditional.

/removeAllConditionals

/removeAllFileFilters

Removes all file filters from the list of file filters OutputDebugString Checker will use to check which files and directories to ignore when checking for function call usage.

This option should be used before any use of /addFileFilter.

/removeAllFileFilters

/removeAllFunctions

Removes all functions from the list of functions OutputDebugString Checker will check.

This option should be used before any use of /addFunction.

/removeAllFunctions

/reportConditional

OutputDebugString Checker will only report functions that are conditionally compiled inside a define specified by the settings

/reportConditional:On|Off

Example: /reportConditional:On
Example: /reportConditional:Off

/reportNonConditional

OutputDebugString Checker will only report functions that are not conditionally compiled inside a define specified by the settings

/reportNonConditional:On|Off

Example: /reportNonConditional:On
Example: /reportNonConditional:Off

/reset

Reset all settings to the default values for this command line search.

You should put this first on your command line to create a known default state prior to setting values using the command line.

/reset

9.2 Usage Reference

Search

/dir

Specifies the directory hierarchy that will be scanned by OutputDebugString Checker.

/dir directory

Example: /dir e:\om\c\

Report

/reportConditional

OutputDebugString Checker will only report functions that are conditionally compiled inside a define specified by the settings

/reportConditional:On|Off

Example: /reportConditional:On
Example: /reportConditional:Off

/reportNonConditional

OutputDebugString Checker will only report functions that are not conditionally compiled inside a define specified by the settings

/reportNonConditional:On|Off

Example: /reportNonConditional:On
Example: /reportNonConditional:Off

Conditionals

/addConditional

Adds a conditional to the list of conditionals OutputDebugString Checker will check.

/addConditional conditional

Example: ./addConditional _DEBUG

/removeAllConditionals

Removes all conditionals from the list of conditionals OutputDebugString Checker will use to check when checking for function call usage.

This option should be used before any use of /addConditional.

/removeAllConditionals

Functions

/addFunction

Adds a function to the list of functions OutputDebugString Checker will check.

/addFunction functionName

Example: ./addFunction myDebuggingFunction

/removeAllfunctionss

Removes all functions from the list of functions OutputDebugString Checker will check.

This option should be used before any use of /addFunction.

/removeAllFunctions

File Filters

/addFileFilter

Adds a filename or directory to the list of file filters OutputDebugString Checker will examine when checking for function call usage.

/addFileFilter filename
/addFileFilter directory

Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\tlcExample.cpp
Example: /addFileFilter e:\om\c\threadLockChecker\tlcExample\

/removeAllFileFilters

Removes all file filters from the list of file filters OutputDebugString Checker will use to check which files and directories to ignore when checking for function call usage.

This option should be used before any use of /addFileFilter.

/removeAllFileFilters

Line Pragma

/linePragmaEnable

Enables or disables the ignoring of locks if a line pragma is found in a comment on the same line as the function call.

If you use this option to enable line pragmas you must use **/linePragma** to specify a line pragma to use. If you don't specify a line pragma this option does nothing.

/linePragmaEnable:On|Off

Example: /linePragmaEnable:On
Example: /linePragmaEnable:Off

/linePragma

Specifies the line pragma to use if line pragmas are enabled.

/linePragma pragma

Example: /linePragma SVL_TLC_PRAGMA

Example usage of the pragma in code:

```
            OutputDebugString(_T("NVIDIA Properties Dump\r\n"));          //
SVL_ODS_PRAGMA
```

Miscellaneous

/loadSettings

Specifies a file to load that contains settings that will override the current settings for this command line search.

/loadSettings filename

Example: /loadSettings e:\tests\test1\testSettings.tlc

/output

Specifies a filename to write the results of the command line search.

If no errors are found any file with this name will be deleted.

If errors are found a text file will be written containing error information. Error information will be listed one error per line.

/output filename

Example: /output e:\tests\test1\testResults.txt

/reset

Reset all settings to the default values for this command line search.

You should put this first on your command line to create a known default state prior to setting values using the command line.

/reset

9.3 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 **outputDebugStringChecker.exe**. You will need to provide the full path to outputDebugStringChecker.exe, or add the path to outputDebugStringChecker install directory to your $PATH.

Example 1

outputDebugStringChecker.exe /dir e:\om\c\ /output e:\test.txt

/dir e:\om\c

Scan the directory e:\om\c\

/output e:\test.txt

Write the output to e:\test.txt. If there are no functions detected there will be no output file.

Example 2

outputDebugStringChecker.exe /dir e:\om\c\ /output e:\test.txt /removeAllFunctions /addFunction myDebuggingFunction

/dir e:\om\c

Scan the directory e:\om\c\

/output e:\test.txt

Write the output to e:\test.txt. If there are no functions detected there will be no output file.

/removeAllFunctions

Remove all existing functions definitions.

/addFunction myDebuggingFunction

Add one function definition so that OutputDebugString Checker is checking for just the function **myDebuggingFunction**

Example 3

outputDebugStringChecker.exe /loadSettings e:\tests\test1\settings.tlc /output e:\test.txt

/loadSettings e:\tests\test1\settings.tlc

Load settings from the file e:\tests\test1\settings.tlc.

/output e:\test.txt

Write the output to e:\test.txt. If there are no errors there will be no output file.

Example 4

outputDebugStringChecker.exe /loadSettings e:\tests\test1\settings.tlc /output e:\test.txt /dir e:\om\c

/loadSettings e:\tests\test1\settings.tlc

Load settings from the file e:\tests\test1\settings.tlc.

/output e:\test.txt

Write the output to e:\test.txt. If there are no errors there will be no output file.

/dir e:\om\c

Scan the directory e:\om\c\