

Sysinternals DebugView

Copyright © 1999-2004 Mark Russinovich
Sysinternals - www.sysinternals.com

DebugView is an application that lets you monitor debug output on your local system, or any computer on the network that you can reach via TCP/IP. It is capable of displaying both kernel-mode and Win32 debug output generated by standard debug print APIs, so you don't need a debugger to catch the debug output your applications or device drivers generate, and you don't need to modify your applications or drivers to use non-Windows debug functions in order to view its debug output.

License

You may not redistribute *DebugView* in any form without the express written permission of Mark Russinovich. If you wish to redistribute *DebugView*, please contact licensing@sysinternals.com.

Capabilities

DebugView works on Windows NT 4.0, Windows 2000, Windows XP, Windows Server 2003, Windows 95, Windows 98 and Windows Me.

Note: if you want to run *DebugView* on Windows 95 you must install the [WinSock2 update](#), available for free download from Microsoft's Web site.

Under Windows 9x/Me *DebugView* can capture output from the following sources:

- Win32 **OutputDebugString**
- Win16 **OutputDebugString**
- Kernel-mode **Out_Debug_String**
- Kernel-mode **_Debug_Printf_Service**

Under Windows NT and Win2k *DebugView* can capture:

- Win32 **OutputDebugString**
- Kernel-mode **DbgPrint**
- All kernel-mode variants of **DbgPrint** implemented in Windows XP and .NET Server

DebugView also extracts kernel-mode debug output generated at the time of a crash from crash dump files if *DebugView* was capturing output at the time of the crash.

Starting DebugView

Simply execute the *DebugView* program file (dbgview.exe) and *DebugView* will immediately start capturing debug output. Note that if you wish to capture kernel-mode debug output under Windows NT/2K, you must have the “load driver” privilege.

Menus, hot-keys, or toolbar buttons can be used to clear the window, save the monitored data to a file, search output, and change the window font. In addition, you can toggle on and off capture of kernel or Win32 debug output.

As events are printed to the output, they are tagged with a sequence number. If *DebugView*'s internal buffers are overflowed during extremely heavy activity, this will be reflected with gaps in the sequence number.

Each time you exit *DebugView* it remembers the position of the window, the widths of the output columns, the font selection, configured filters, and the time-stamp mode.

Command-line Options

DebugView supports several command-line options that let you modify its behavior when it starts. Several are relevant when starting *DebugView* as a client on a system that will send debug output across the network to a *DebugView* instance that displays the output on another computer, and are described in the [Remote Monitoring](#) section. However, others modify the behavior of *DebugView* when you run it to display output, and are useful if you want to execute *DebugView* from a batch file or logon script and want it to capture debug output as soon as it starts. You can have *DebugView* display all of its command-line options by using the */?* option.

Here are the command-line options supported when you run *DebugView* in non-client mode:

```
debugview [/f] [/t] [/l Logfile [/a] [[/m nnn [/w]] | [/n [/x]]] [/h nn]] [Logfile]
```

The */f* option has *DebugView* skip the filter confirmation dialog when filters were active the previous execution.


The */t* option has *DebugView* launch into the system tray, rather than as a window. This has *DebugView* capture debug output as soon as it starts while not taking up screen real-estate. *DebugView's* tray behavior is further described in the [Running in the Tray](#) section.

The */l* option directs *DebugView* to begin writing output to the indicated logfile as soon as *DebugView* executes. The */m* option allows you to specify a size limit (in MB) for the log file, and the */a* option has *DebugView* append to the logfile if it already exists, rather than overwrite it and */w* has the log file wrap when it reaches the maximum size you specify. The */n* switch has *DebugView* create a new log file, named with the date, each day. If you include */x* with */n* the display clears when a new log file is created.


Finally, the */h* switch controls the history depth, which is the count of most recent output lines shown in the *DebugView* display. These options correspond to the logfile commands available through menu items when *DebugView* is running, which are described in [Saving and Print](#).

Capturing Debug Output

Global Capture

You control *DebugView*'s global capture mode by toggling capture-on and capture-off with the  toolbar button, the **Capture|Capture Events** menu entry, or the Ctrl+E hot-key sequence. *DebugView* does not capture any data when its capture-mode is off. The state of Win32 capture and kernel capture determine what kind of debug output (if any) is captured when the global capture mode is on.


Capturing Win32 Debug Output

If you specify, *DebugView* will register to receive and print debug output generated by Win32 programs that call OutputDebugString. The  toolbar button, the **Capture|Capture Win32** menu item, and the Ctrl+W hot-key sequence can toggle this capture on and off. If the Win32 PID options is set (under **Options|Win32 PIDs**) then information identifying processes that generate Win32 debug output is prefixed to each line of Win32 debug output. If you are running *DebugView* on Windows NT/2K, then the process ID of the processes are prefixed in brackets to each line of Win32 debug output. If the option is set and you are running on a Win9x system, then the process name is prefixed in brackets to the output.

If you run *DebugView* in a remote logon session of Windows 2000 Terminal Services, *DebugView* adds a **Capture Global Win32** menu item to the **Capture** menu. Whereas the **Capture Win32** menu item and associated toolbar button enable and disable capture of debug output in *DebugView*'s local logon session, the **Capture Global Win32** menu item lets you enable and disable the capture of debug output that is generated in the console (global) session. Win32 services run in the console session, so this feature lets you capture the output that services generate even when you are running *DebugView* in another logon session.

Capturing Kernel-Mode Debug Output


You can configure *DebugView* to capture kernel-mode debug output generated by device drivers and/or the Windows kernel by using the

Capture|Capture Kernel menu selection,  toolbar button, or the Ctrl+K hot-key. Process IDs are not reported for kernel-mode output since such output is typically not process-context related.

On Windows NT/2K, kernel-mode capture is only possible if the user account in which you run *DebugView* has the "load driver" privilege. If the account does not have this privilege then *DebugView* disables the kernel-mode capture and pass-through mode toolbar buttons and menu items.

Under Windows NT/2K, Win32 debug output and kernel-mode debug output originate with two different sources. Therefore, *DebugView* captures the different outputs into two separate buffer pools, and merges the outputs in the display window according to their relative sequence numbers. While this means that the order of kernel-mode and Win32 output is correctly represented, the update of such information may not be sequential i.e. *DebugView* may display a number of Win32 debug messages, and shortly after merge in kernel-mode debug messages that have interleaved sequence numbers. This is the reason that sequence numbers are represented as 8-digit numbers: the display's listview auto-sorting feature is used by *DebugView* to order merged kernel-mode/Win32 output.

Pass-Through Mode

DebugView can be configured to pass kernel-mode debug output to a kernel-mode debugger or to swallow the output. The pass-through mode is toggled with the **Capture|Pass-Through** menu selection or  toolbar button. The pass-through mode allows you to see kernel-mode debug output in the output buffers of a conventional kernel-mode debugger while at the same time viewing it in *DebugView*.

Inserting Comments

You can insert comments into the output log by selecting **Edit|Append Comment**. Comments insert into the currently viewed output. Type comments into the dialog followed by the return key and dismiss the dialog when you are done entering comments.


Clearing the Display

An application that generates debug output can cause the *DebugView* display to clear by printing the string "DBGVIEWCLEAR".


Searching, Filtering, and Limiting Output

DebugView has several features that can help you zoom-in on the debug output you are interested in. These capabilities include searching, filtering, and limiting the number of debug output lines saved in the display.


Clearing the Display

To reset the output window, use the **Edit|Clear Display** menu item,  toolbar button, or Ctrl+X hot-key sequence. This also causes the sequence number to be reset to 0.

Searching

If you want to search for a line containing text of interest you use the find dialog. The find dialog is activated with the Ctrl+F hot-key sequence, the **Edit|Find** menu entry, or the  toolbar button. If the search you specify matches text in the output window, *DebugView* will highlight the matching line and turn off the display's auto-scroll in order to keep the line in the window. To repeat a successful search, use the F3 hot-key.

Filtering

Another way to isolate output that you are interested in is to use *DebugView's* filtering capability. Use the **Edit|Filter/Highlight** menu item,  toolbar button, or Ctrl-L hot-key to activate the filter dialog. The dialog contains two edit fields: include and exclude. The include field is where you enter substring expressions that match debug output lines that you want *DebugView* to display, and the exclude field is where you enter text for debug output lines that you do not want *DebugView* to display. You can enter multiple expressions, separating each with a semicolon (;). Do not include spaces in the filter expression unless you want the spaces to be part of the filter. Note that the filters are interpreted in a case-insensitive manner, and that you should use '*' as a wildcard.

As an example, say you want *DebugView* to display debug output that contains either "error" or "abort", but want to exclude lines that contain either of those strings and the word "gui". To configure *DebugView's* filters for this you enter "error;abort" for the include filter and "gui" for the

exclude filter. If you wanted to have *DebugView* show only output that has "MyApp:" at the start of the output line and "severe" at the end, you could use a wildcard in the include filter: "myapp:*severe".

Highlighting


DebugView also has another type of filtering: highlighting. If you want output lines that contain certain text to be highlighted in the *DebugView* output window, enter a highlight filter. *DebugView* implements support for up to five different highlight filters, each with its own foreground and background color settings. Use the filter drop-down in the highlight filter area of the filter dialog to select which highlight filter you want to edit. Use the same syntax just described for include and exclude filters when defining a highlight filter.

To change the colors used for the foreground and background of highlighted lines, click on the Colors button when you have selected the highlight filter you wish to modify. You will be asked to pick colors from a palette, and your choices will be remembered by *DebugView* from run to run.

Saving and Restoring Filters

Use the **Load** and **Save** buttons on the filter dialog to save and restore filter settings, including the include, exclude and highlighting filters, as well as the highlighting colors settings.

History-Depth

A final way to control *DebugView* output is to limit the number of lines that are retained in the display. You use the **Edit|History-Depth** menu item,  toolbar button, or the Ctrl+H hot-key sequence to activate the history-depth editor. Enter the number of output lines you want *DebugView* to retain and it will keep only that number of the most recent debug output lines, discarding older ones. A history-depth of 0 represents no limit on output lines retained.

You do not need to use the history-depth feature to prevent all of a system's virtual memory from being consumed in long-running captures.

DebugView monitors system memory usage and will go into a low-memory state when it detects that memory is running low. *DebugView's* low-memory state consists of it not capturing further debug output until the low-memory condition is no longer in effect.

Saving, Printing and Logging


DebugView lets you both save and print captured debug output.

Saving Output



You can save the contents of the *DebugView* output window as a text file (.log extension) using the **File|Save** or **File|Save As** menu items, or the Ctrl+S hot-key sequence.

Using **Edit|Copy** or the Ctrl+C hot-key sequence you can copy the debug output contained within selected output lines to the clipboard.

Logging to a File

To have *DebugView* log output to a file as it displays it, use the **File|Log to File** or **File |Log to File As** menu items, the  toolbar button, or the Ctrl+O hot-key sequence. Log file settings you specify include the name of the log file, the maximum size it should be allowed to grow, and whether or not *DebugView* should restart the log or append to it if the file specified already contains output. If you select the **wrap** option then *DebugView* will wrap around to the beginning of the file when the file's maximum specified size is reached.

If you select the Create New Log Every Day option then *DebugView* will not limit the size of the log file, but will create a new log file every day that has the current date appended to the base log file name you enter.

When logging is active the log file toolbar button will look like . To stop logging simply select the toolbar button or the **File |Log to File** menu item. If the log file's maximum size is reached logging to the file stops and the logging toolbar button changes to .

If you are monitoring debug output from multiple remote computers and enable logging to a file, all output is logged to the file you specify. Ranges of output from different computers are separated with a header that indicates the name of the computer from which the subsequent records were recorded.

Printing Output

You can use **File|Print** or **File|Print Range** to print the contents of the display to a printer. Choose **Print Range** if you only want to print a subset of the sequence numbers displayed, or **Print** if you want to print all the output records. The Ctrl+P hot-key sequence corresponds to **File|Print**.

Using the **Print Range** dialog you can also specify whether or not sequence numbers and timestamps will be printed along with the debug output. Omitting these fields can save page space if they are not necessary. The settings you choose are used in all subsequent print operations.

In order to prevent wrap-around when output lines are wider than a page, consider using landscape mode instead of portrait when printing.


Loading Output

Use the **File|Open** menu item to load a previously saved DebugView log file into the output window.

Options

There are a number of options that let you adjust several characteristics of *DebugView*, including the way that it behaves and looks.

Timing Format

DebugView displays time stamps of captured debug output in one of two formats: as clock-time (the time of day), or as relative time. When displaying relative time *DebugView* represents the time of a debug output record as the difference between its timestamp and the timestamp of the first record in the display. This mode is helpful when you debug timing-related problems. Use the **Options|Clock** Time menu item,  toolbar button, or Ctrl+T hot-key sequence to toggle between clock time and relative time modes.


When *DebugView* is in clock-time mode you can select the **Options|Show Milliseconds** menu item to have *DebugView* show timestamps that include millisecond resolution.

Force Carriage Returns

The default behavior of *DebugView* is to buffer output strings in an internal buffer *DebugView* maintains until a carriage-return is encountered or the buffer overflows. This allows applications and drivers to build output lines with multiple invocations of debug output functions.

Select **Options|Force Carriage Returns** to cause *DebugView* to display every string passed to a debug output function on a separate output line, regardless of whether the string is terminated with a carriage return.

Auto Scroll

Use the **Options|Auto Scroll** menu item,  toolbar button, or Ctrl+A hot-key sequence to toggle *DebugView* between auto-scroll and non-auto scroll modes. When in auto-scroll mode *DebugView* will always keep the most recent debug output visible in the display window.

Hiding the Toolbar

You can gain more display space by hiding the *DebugView* toolbar. Use

the **Options|Hide Toolbar** menu item or Ctrl+B hot-key sequence to toggle the toolbar's presence. *DebugView* will remember the toolbar state when you exit it and restore the same state the next time you start it.

Win32 PIDS

Setting this option using the **Options|Win32 PIDs** menu item will cause *DebugView* to prefix Win32 debug output with either the process ID (Windows NT/2K) or the process name (Windows 9x) of the process that generated the output. Deselecting this option can save screen space if you are not interested in what process generates Win32 output.

Changing the Font

Use the **Edit|Font** menu entry to open a font-selection dialog where you can choose a font that *DebugView* will use in its output window.

Always on Top

To keep *DebugView* as the top-most window on the desktop, use the **Options|Always On Top** menu item. Selecting the menu item a second time will toggle off the always-on-top mode.

Running in the Tray

Running *DebugView* in the system tray is useful if you want it to capture debug output but do not it to take up space on the desktop or task bar. You minimize *DebugView* to the system try by selecting the **Edit|Minimize to Tray** menu item, which both changes the menu item to **Edit|Minimize to Task Bar** and has *DebugView* appear as an icon on the tray. To reactivate *DebugView* from the tray you double-click on its tray icon. Subsequently minimizing *DebugView* by clicking on its window minimize button will minimize it to the tray. To minimize *DebugView* to the task bar, select **Edit|Minimize To Task Bar**, after which the minimize button will function as normal.

When *DebugView* is in the tray its icon is colored if global capture is enabled and black-and-white if global capture is disabled. The right-click context menu for the *DebugView* tray icon is a copy of the *DebugView* **Capture** menu, which allows you to enable and disable global capture, Win32 capture, and kernel-mode capture.

Logging at Boot Time (WinNT/2K/XP Only)

Under Windows NT/2K/XP *DebugView* can capture kernel-mode debug output generated during the boot process. To have it do so, select the **Capture|Log Boot** menu item, which is enabled when the local computer is Windows NT/2K and the *DebugView* window is connected to the local computer. When boot logging is enabled, *DebugView* buffers up to 1MB of debug output beginning at the earliest point in the system's next boot process. You can view the buffered debug output by connecting *DebugView* to the system, at which time the buffering ceases.

Crash Dumps (WinNT/2K/XP Only)

Under Windows NT/2K/XP you can configure the system to save a dump of physical memory when the operating system crashes. Using this crash dump facility *DebugView* allows you to view any debug output your kernel-mode driver made up to the time of a crash. If your driver is sufficiently instrumented with debug output, then this feature permits users that experience a crash using your driver to send you a debug output file instead of an entire memory dump. You must be capturing kernel-mode debug output with *DebugView* at the time of crash for this option to work.

Use the **Edit|Process Crash** menu item to select a crash dump file for *DebugView* to analyze. *DebugView* will process the file, looking for its debug output buffers. If it finds debug output in the crash dump *DebugView* will prompt you for the name of the log file where it should save the output. You can load saved output files into *DebugView* for viewing.

Remote Monitoring Startup

DebugView has advanced remote monitoring capabilities that allow you to view debug output generated on remote systems from a central location. The remote systems must be accessible via TCP/IP. *DebugView* lets you monitor multiple remote systems simultaneously, using a hot-key or a menu selection to switch between them. If both the computer you are running the *DebugView* GUI on (the server) and the system you want to monitor (the client) are running Windows NT/2K, and they are in the same Network Neighborhood, then *DebugView* will automatically install its client software on the client. For all other combinations you must manually install and start *DebugView's* client software on the client.

Manual Client Startup

If either the server or the client is running Windows 9x, or the server and client are not mutually accessible via the Windows Network Neighborhood, then you must manually start the *DebugView* client on the client computer. To do this, run the *DebugView* program on the client and specify "/c" as a command-line argument:

```
dbgview /c [/t] [/s] [/e] [/g]
```

The *DebugView* client window will appear and indicate that it is waiting for a connection from the *DebugView* server.

After you have started the *DebugView* client use the **Computer|Connect** menu item or Ctrl+R hot-key sequence of the *DebugView* server to open a computer connection dialog. In the dialog enter the name or IP address of the client computer. If the client computer is in the server's Network Neighborhood you can also use the browse button in the dialog to open a view of the Network Neighborhood and visually select the client computer.

If you want to run the client in a "headless" mode, specify "/s" (silent) in addition to the "/c" command-line argument when you start the *DebugView* client. This will cause the *DebugView* client to not display a window, and to remain active until the current user logs out, silently connecting with and disconnecting from *DebugView* servers.

Use the “/e” option when starting the client if you want it to notify you when server connections break. When a server connection is broken and this switch is specified you must close the notification window before the client will accept further connections.

The “/t” option has the *DebugView* client run in the tray. The client presents a gray tray icon when there's no connection to a server and a colored icon when a server is connected. You can open the client window by double clicking the tray icon and store it back in the tray by minimizing the client window.

If you are running *DebugView* from a non-console login on a system with Terminal Services you can direct the *DebugView* client to capture global (console) debug output with the /g switch.

If you specify “/?” *DebugView* will tell you its supported command-line options.

Automatic Client Startup

Automatic startup is not supported on the Alpha.

If both the client and server are running Windows NT/2K and are in the same Network Neighborhood, there is no need for you to install the *DebugView* client on the client computer. Instead, specify the client computer name or address in the connection dialog as you would if you were connecting to a manually started *DebugView* client, and *DebugView* will automatically install and start the *DebugView* client on the client computer. When you disconnect from the client *DebugView* uninstalls its client software for you. In case you want to clean up client files yourself after a non-graceful exit of the server, the files *DebugView* installs on the client are placed in <winnt>\system32 and include dbgvsvc.exe and dbgv.sys.

The *DebugView* server will always attempt an automatic install, and if that fails it falls back on trying to connect to a manually installed client.

Managing Connection Views

When a remote capture session is established *DebugView* creates a new computer view for the session. The active computer view is the one that has captured output displayed in the *DebugView* GUI, and is identified on the *DebugView* title bar. To switch from one computer view to another select the desired view, which is listed by computer name, in the **Computer** menu. Alternatively, you can use Ctrl+Tab to cycle through the computer views.

The state of global capture, Win32 debug capture, kernel capture, and pass-through for a newly established remote session are all adopted from the current settings of the local view (the view of the computer on which the *DebugView* is executing). Changes you make to these settings only apply to the active computer view.

Disconnecting a Remote Session

When you are through capturing debug output from a remote system, make the view for the computer from which you want to disconnect the active view and then use the **Computer|Disconnect** menu entry to close the session.

When you exit *DebugView* it saves the state of the local view, including the width of the display columns, and *DebugView* applies those settings the next time you start it.

Managing Multiple Windows

DebugView allows you to open multiple *DebugView* windows on the same computer, allowing you to capture debug output from different computers into different windows. This is an alternative to connecting to multiple computers from the same *DebugView* window, and is desirable when you wish to simultaneously view different output sources.

By default, when you start the first *DebugView* window on a computer it connects with the local computer. This means that it captures and displays any debug output generated on the computer. You can open a second instance of *DebugView* either by starting it again, or by selecting the **File|New Window** menu entry.

You can use the **Computer|Connect Local** menu entry to connect *DebugView* to the local computer, and choose the **Computer|Disconnect** menu entry to disconnect from the local computer when it is selected as the active computer view within a *DebugView* window. Note that only one *DebugView* instance can be connected to any computer at a given time.

If you start a new *DebugView* window by executing the program again the configuration settings *DebugView* uses reflect those of the last *DebugView* window that was closed. If you start a new *DebugView* window using the **File|New Window** menu entry, the configuration settings are adopted from the window in which you select the menu item.

Reporting Problems

If you encounter a problem while running *DebugView*, please visit the [Sysinternals](http://www.sysinternals.com) web site (www.sysinternals.com) to see if an update has been released that might correct the bug. If the problem has not been fixed, please submit a thorough report of the problem, including information on your system configuration and details on how to reproduce the problem, to:

mark@sysinternals.com