

TOTALVIEW NEW FEATURES GUIDE



VERSION 8.6

Copyright © 2007–2008 by TotalView Technologies. All rights reserved

Copyright © 1998–2007 by Etnus LLC. All rights reserved.

Copyright © 1996–1998 by Dolphin Interconnect Solutions, Inc.

Copyright © 1993–1996 by BBN Systems and Technologies, a division of BBN Corporation.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise without the prior written permission of TotalView Technologies.

Use, duplication, or disclosure by the Government is subject to restrictions as set forth in subparagraph (c)(1)(ii) of the Rights in Technical Data and Computer Software clause at DFARS 252.227-7013.

TotalView Technologies has prepared this manual for the exclusive use of its customers, personnel, and licensees. The information in this manual is subject to change without notice, and should not be construed as a commitment by TotalView Technologies. TotalView Technologies assumes no responsibility for any errors that appear in this document.

TotalView and TotalView Technologies are registered trademarks of TotalView Technologies. TVD is a trademark of TotalView Technologies.

TotalView uses a modified version of the Microline widget library. Under the terms of its license, you are entitled to use these modifications. The source code is available at <http://www.totalviewtech.com/Products/TotalView/developers>.

All other brand names are the trademarks of their respective holders.

Contents



New Features: Versions 8.0–8.6

New Platforms and Compilers for 8.6	1
8.6 Changes	2
ReplayEngine	2
TotalView Changes.....	2
Changes in Previous Version 8 Releases	4
New and Changed Features	5
Version 8.4 Features	5
Version 8.3 Features	5
Version 8.2 Features	7
Versions 8.0 and 8.1 Features	8
Other Features	9

New Features:

Versions 8.0–8.6



This document contains information about changes made to TotalView for versions 8.0.0 through 8.6. While this information let you know what changes have occurred, It doesn't completely describe these changes. Instead, you'll find descriptions for most of the changes within the *TotalView Users Guide* and the *TotalView Reference Guide*.

TotalView has many features and it gives you a great number of tools for finding your program's problems. An easy way to get acquainted with these features is to subscribe to the "Tip of the Week". If you subscribe to this mailing list, you'll receive an email message every week (or most weeks) that tells you something about TotalView and debugging.

- All of the tips are archived on our web site at <http://www.totalviewtech.com/Documentation/Tips/index.html>.
- If you like what you see, you can subscribe by going to <http://www.totalviewtech.com/mojo/mojo.cgi>.

New Platforms and Compilers for 8.6

TotalView supports new version of compilers and environments. New to TotalView are C/C++ compilers for the IBM Cell Broadband Engine, GNU Fortran from Red Hat, and the Sony BCU-100 Zego.

8.6 Changes

ReplayEngine

The ReplayEngine is a separately licensed product that is integrated with TotalView. It is available on Linux-86 and Linux-86-64 machines.

- ReplayEngine records the changes to program state as they happen.
- ReplayEngine replays previously executed commands. ReplayEngine commands are added to the TotalView toolbar. Generally, these new commands let you specify which previously executed line in your program you want to examine. These new commands are analogous to Next, Step, Out, and Run To. They differ in that they move into the program's history. That is, entering replay mode is done by pressing a single button. The commands behind these buttons are located at the new **Instrumentation** entry added to the tool bar. (All memory commands have also been moved to **Instrumentation**.)
- When you are in replay mode, you can step use ReplayEngine commands to move through your program's assembly code.
- Changing back to record mode is as simple as pressing the **Live** tool bar button. Reentering replay mode is just a button press.
- Multithreaded codes replay in precisely the same sequence that the threads previously executed. This is especially useful for examining race conditions.
- Breakpoints, watchpoints, and some conditional breakpoints can be used when running forward in replay mode.
- ReplayEngine can be used with the TotalView Memory Debugger.

Information on running ReplayEngine is at <http://www.totalviewtech.com/Documentation>. This information is also contained in the online help that comes with TotalView. This help is available even if you haven't purchased a ReplayEngine license.

TotalView Changes

Remote Display

TotalView can open a window on your machine that you will display TotalView executing on a remote system. We provide installers for Windows running XP or Vista, Linux-x86 and Linux-x86-64.

While Remote Display can only run on these operating systems, the remote TotalView can execute upon any platform that TotalView supports.

Information on running Remote Display is at

<http://www.totalviewtech.com/Documentation>

tvscript

TotalView and the Memory Debugger can run unattended if you use the **tvscript** shell command. This is called batch debugging because, like all batch programs, you do not need to use them interactively. In addition, you

can invoke **tvscript** using **cron** to schedule debugging in the evening or on weekends. In this way, you'll have reports waiting for you in the morning.

The commands that **tvscript** executes can be entered in two ways. The first is as command-line options. The second—and this is the preferred method—is by creating a file and then have **tvscript** execute the commands in this file. The reason that this is the preferred method is that you can create Tcl callback functions that are called when an event occurs within your program. These callback functions can also include CLI commands.

Chapter 4 of the *TotalView Reference Guide* explains how to use the **tvscript** command. You can obtain this document by going to:

<http://www.totalviewtech.com/Documentation>

Other Changes

The two major changes to TotalView at this release are:

- **Debug** is added to the TotalView menubar in you are running on Linux-x86 and Linux-x86-64. All memory commands are now within **Debug**.
- The current line is highlighted in yellow. If you have purchased a Replay license, an orange highlight line shows where you've gone back to. A separate marker shows the PC that existed when you entered replay mode.
- The **dhistory** command lets you invoke ReplayEngine from within the CLI. The **spurs** command displays information on spurs library use.
- Options supporting ReplayEngine are added to **dattach**, **dload**, and stepping commands such as **dstep**, **dnext**, **duntil**, etc.
- Options to the **dload** command let you start MPI programs. These options are **-mpi**, **-nodes**, **-starter_args**, **-np**, **-procs**, and **-tasks**.
- The **Process > Startup Parameters** dialog is rearranged and now contains options for enabling memory debugging and ReplayEngine. This window comes up automatically when you start TotalView using a program's name as an argument. To suppress this box, select the check box at the bottom of the Parallel page. When invoking TotalView this way, you can suppress this display using the **-no_show_startup_parameters** command-line option.
- Other command-line options added for this release include **-local_interface** (most often used with Blue Gene), **-memory_debugging**, and **-replay**.
- New variables added at this release are **TV::ask_on_cell_spu_image_load**, **TV::cell_spu_image_ignore_regexp**, **TV::cell_spu_images_stop_regexp**, **TV::cell_spurs_jm_name**, **TV::cell_spurs_kernel_dll_regexp**, **TV::cell_spurs_ss_name**, **TV::cell_spurs_tm_name**, **TV::data_format_int128**, **TV::local_interface**, and **TV::GUI::heap_summary_refresh**.

Changes in Previous Version 8 Releases

8.5 Changes TotalView now supports the IBM Cell Broadband Engine.

8.4.1 Changes This release has updated the compilers TotalView supports. Consult the *TotalView Platforms and System Requirements Guide*. for more information.

8.4 Changes TotalView now supports Apple Mac OS X 10.5 (Leopard).

8.3 Changes New operating system versions include:

- Apple OS X 10.4.5, 10.4.8, and 10.4.9
- Fedora Core 7
- Ubuntu 6.06

As always, we have added support for new versions of existing compilers and parallel runtime environments. Consult the *TotalView Platforms and System Requirements Guide*. for more information.

8.2 Changes TotalView 8.2 has added support for the following systems and compilers:

- SiCortex Support
SiCortex delivers ultra-low power, high-performance Linux computers for the HPC market. TotalView will now support debugging parallel applications on the new SiCortex supercomputer.
- Cray XT4 Support and APLs Integration
- Fedora Core 6 Support
- Expanded Mac Support
Preliminary Mac OS X Leopard Support, 64-bit Mac-Intel Support and Mac Universal Binaries Support
- Ubuntu Support
Ubuntu is a community-developed Linux-based operating system for the desktop, laptop, thin client and server. TotalView will support applications developed on this new platform.

You'll find a complete list of supported platforms and compilers in the *TotalView Platforms and System Requirements Guide*.

New and Changed Features

Version 8.4 Features

This section lists the changes made for version 8.3 of TotalView.

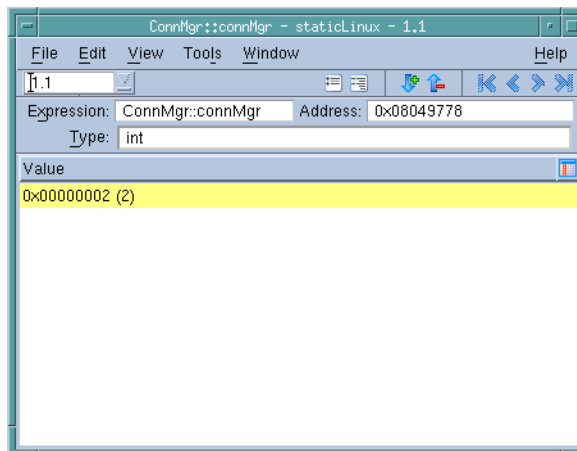
- If you have more than one TotalView license, you can control which kind of license TotalView uses by adding one of the following command-line options: `-team`, `-noteam`, `-teampplus`, `-noteampplus`, `-ent` or `-noent`. For more information, see Chapter 6 of the *"TotalView Reference Guide."*
- The TotalView Memory Debugger can now write light-weight memory debug files when an event occurs. These files are similar to the memory debugging files (`.mdbg`) files that you can write using the **File > Export** command. They differ in that they are designed to be written when the event occurs and in such a way that the program's behavior is minimally disturbed. These files are described in the Chapter 3 of the *"Debugging Memory Problems with TotalView"* Guide.
- Improved support for C++ templates.
- Improved support for Fortran modules on Apple Mac OS X.

Version 8.3 Features

This section lists the changes made for version 8.3 of TotalView.

- Improvements to the way TotalView launches MPI programs let you use TotalView with virtually every MPI library, even with those that were not configured for debugging.
- TotalView now highlights changes to values displayed in the **Variable** and **Expression List** windows with a colored background. (See Figure 1 on page 5,)

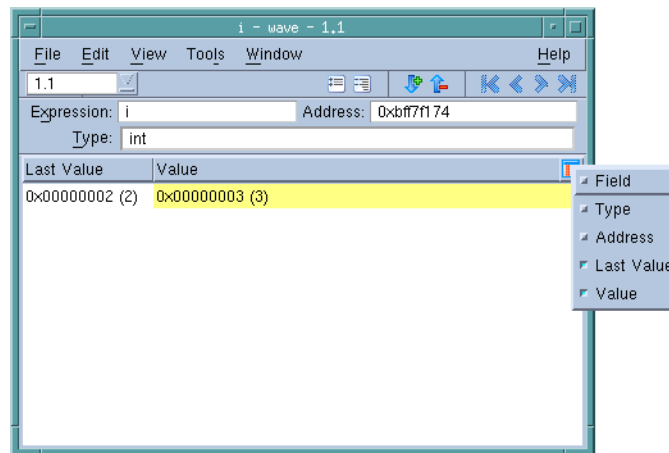
Figure 1: Highlighted Change in the Variable Window



While this figure shows a simple variable, TotalView also highlights changed elements within compound variables such as structures and arrays.

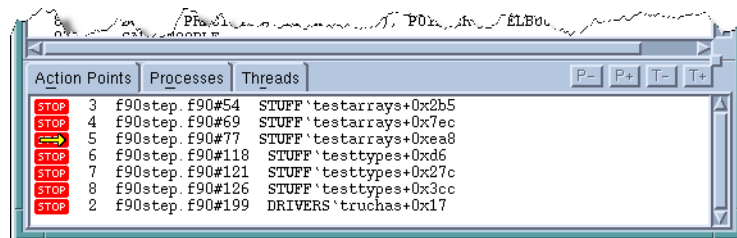
- Values in the **Variable** and **Expression List** windows have a **Previous value** hidden column that you can display. Use the control on the right side of the column headings to display a list of columns that you can display or hide. (See Figure 2 on page 6,)

Figure 2: Last Value Column



- When a process hits a breakpoint, TotalView highlights the breakpoint by placing an arrow over the breakpoint ID in the Action Points pane. (See Figure 3 on page 6,)

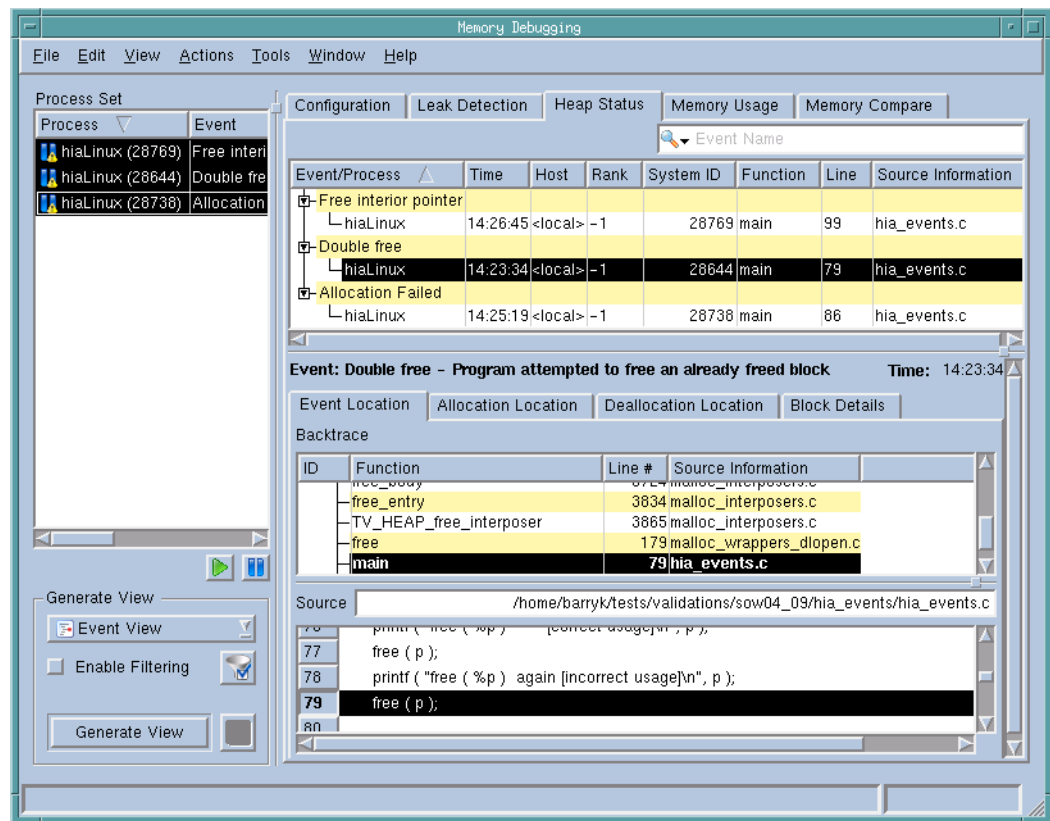
Figure 3: Highlighted Action Point ID



- **View > Show Across** replaces **View > Laminate** in the **Variable** window's menus. This means the commands you will now use are **View > Show Across > Process** and **View > Show Across > Thread**.
- You can now tell TotalView to show a variable across processes or threads by right-clicking on it in the Source Pane, then selecting either **Across Processes** or **Across Threads** from the context menu.
- The **Create Watchpoint** command was added to the Action Points menu. As always, you can create a watchpoint from within the Variable window by selecting **Tools > Create Watchpoint**.
- You can now set a watchpoint upon a variable's memory address by right-clicking on the variable in the Source pane and then selecting **Create Watchpoint** from the context menu.

- You can completely expand or collapse information in the Variable window by selecting an icon in the toolbar. The accelerators for these commands are **Ctrl++** (that's the control key and the + symbol) and **Ctrl+-** (which is the control key and the - symbol).
- TotalView no longer stops by default when your program loads a library.
- You can specify more than one core file on the command line and you can use wildcards in core file names.
- There is a new Events View report within the Memory Debugger Heap Status tab.

Figure 4: Event View



- Within the Memory Debugger's **File > Import Data** dialog box, you can select multiple memory debugging (.mdbg) files.

Version 8.2 Features

This section looks at changes that have occurred within TotalView.

- **Early-Access GUI Installer**
You can now install TotalView from tar files as you've always done or install it using our new graphical installer. We are calling this an early-access release in that we want you to tell us what you think of it and how we can improve it.

- Fortran Parameter Display
TotalView now displays the value of Fortran parameters. Parameters can be used like variables in expressions but could not previously be examined within the debugger.

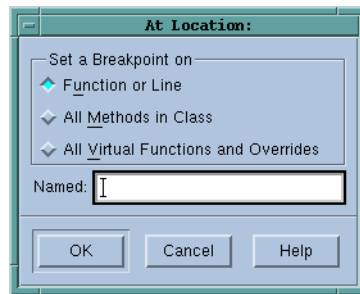
Versions 8.0 and 8.1 Features

Breakpoint Changes

Action points are considerably more powerful. Here is a summary:

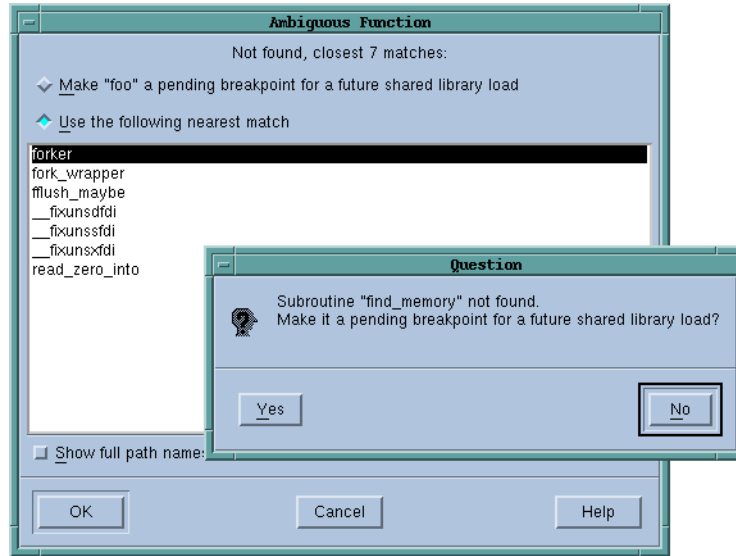
- The **Action Point > At Location** command now has three choices:
 - **Function or Line**: lets you set a line number or a function name. This choice is what occurred in previous TotalView releases.
 - **All Methods in Class**: lets you set a breakpoint on all methods in a class. This can set more than one breakpoint.
 - **All Virtual Functions and Overrides**: lets you set breakpoints on virtual functions and their overrides. This too can set more than one breakpoint.

Figure 5: Breakpoint > At Location Dialog Box



- You can now tell TotalView that a breakpoint will occur in a library that will be loaded later and that TotalView should retain knowledge of this breakpoint. This allows it to be set when the library is read. Previously, TotalView had to create and set breakpoints at the same time. This meant that when you enter a name into the **At Location** dialog box and the name is not yet known, TotalView, displays either an **Ambiguous Function** or a **Question** dialog box. At this time, you can set the breakpoint's status to *pending*. (See Figure 6 on page 9.)
- When you create a barrier breakpoint or change a breakpoint to a barrier point, the same new features are available.
- The CLI **dbreak**, **dbarrier**, and **dlist** commands have been extended to use these features. The argument to these commands can now be a breakpoint expression. Understanding this concept reveals some of the subtleties involved using these new features. These concepts are explained with the **dbreak** pages of the *TotalView Reference Guide*.

Figure 6: Setting Pending Breakpoints

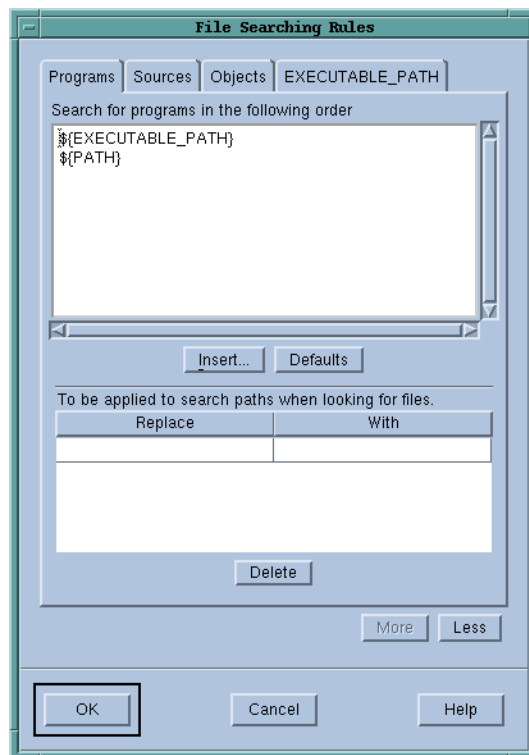


Other Features

This section describes improvements and changes made in many different places in TotalView.

- The way in which you set search paths has changed. (See Figure 7 on page 9.)

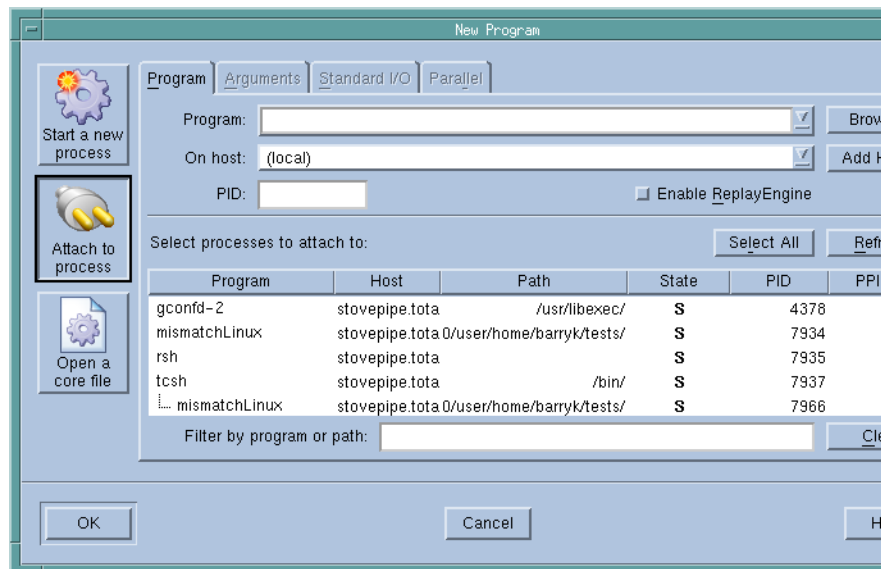
Figure 7: File > Search Path Dialog Box



You have told us that most of the time, you usually do not need to set search paths. If you do, you just need to enter a couple of paths. This is what the old dialog box was designed for. (You can still do this by typing paths directly into the `EXECUTABLE_PATH` tab.) However, when your programs make the transition from modules being developed on your workstation to an place where the work of development teams is brought together, setting search paths was tedious and difficult to get right. This new dialog box, extensively documented in the help, lets you solve this problem.

- The **File > New Program** dialog box has changed. Much of what you will see makes the dialog box more usable. Most notable is the way you attach to processes. The dialog box now lets you select more than one process at a time. The one new feature is that you can now enable memory debugging from this dialog box.

Figure 8: File > New Dialog Box



- The **View > Freeze** command is added to the Variable window. This command tells TotalView that it should freeze the contents of a Variable window. That is, as your program executes and as data values change, the contents of this window does not change. In most cases, you will also create a second Variable Window so that you can see old and new values at the same time.
- The **View > Lock** command is added to the Variable Window. This command tells TotalView that it should not change the address from which the Variable Window is obtaining information.
- New **dheap -compare** CLI command options. This options lets you compare the result of two different memory states.
- New **dkill -remove** CLI command option. Using this option to tell TotalView that, in addition killing the process, it should remove knowledge of the process. This is seldom necessary. However, if you are using

TotalView Team, using this option makes the token used by a process available to another process in your program.

- The following CLI variables were added for this release
 - **TV::env**: sets an environment variable.
 - **TV::bluegene_server_launch_string**: sets the Blue Gene server launch string.
 - **TV::default_sterr_append**: tells TotalView to append **stderr** information to **stdout**.
 - **TV::default_stderr_filename**: tells TotalView to write **stderr** information to a file.
 - **TV::default_stderr_is_stdout**: tells TotalView to write **stderr** information to **stdout**.
 - **TV::default_stdin_filename**: Tells TotalView to read **stdin** information from a file
 - **TV::default_stdout_append**: Tells TotalView to append **stdout** information to a file
 - **TV::default_stdout_filename**: Tells TotalView to write **stdout** information to a file.