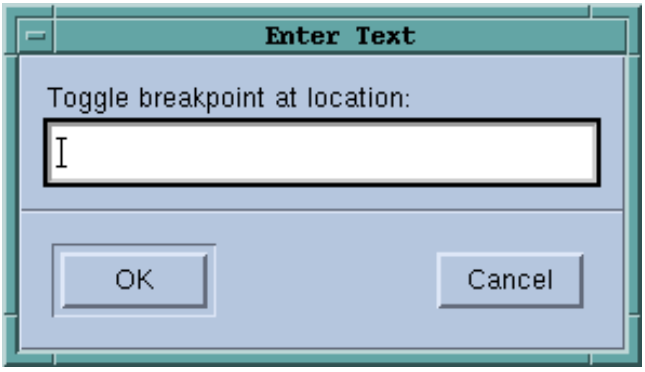
Tip-of-the-Week

Is there a quick way to set a breakpoint on a function?

Use the ***Action Point > At Location*** command.



After typing a function name, TotalView sets a breakpoint on the first executable statement within the function.

Bonus Tip: Shift-Clicking Between Barrier Points and Breakpoints

You can change a breakpoint into a barrier point by shift-clicking on the stop icon. Shift-clicking shifts it back to a breakpoint.

You can find tips that we've already sent out in our **Tip Archive**

Help us improve these tips!

- Tell us if you liked or disliked this tip.
- Is there something you'd like to be a tip?
- Tell us your tips.

Etnus, LLC
www.etnus.com
Phone: (508) 652-7700
documentation@etnus.com

Tip-of-the-Week

ETNUS

Why can't I get a process (or thread) held at a barrier to execute?

Note: *Last week's Tip was also about barriers. Because processes and threads behave similarly, I'll only talk about processes.*

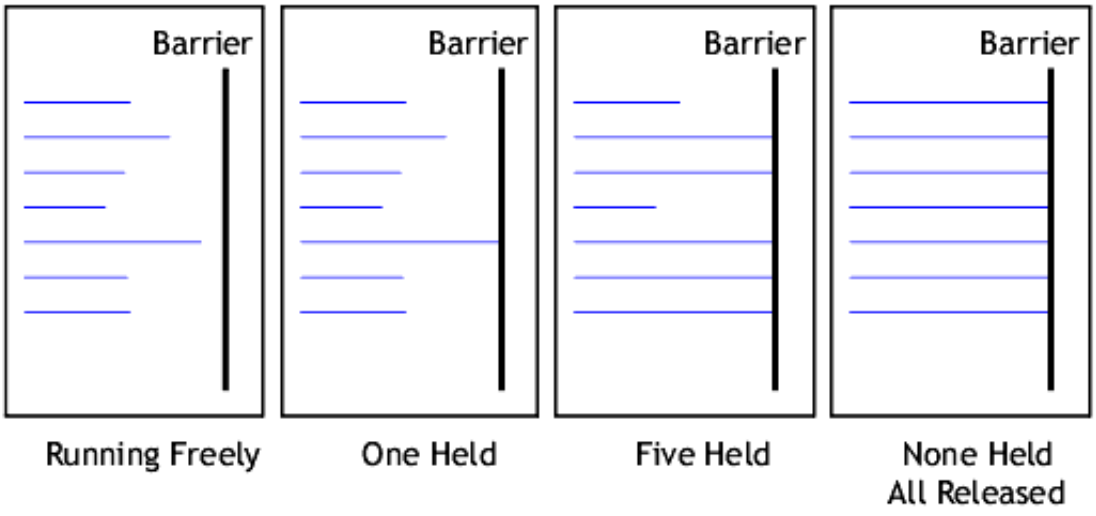
Creating a barrier point tells TotalView that it should *hold* a process when it reaches the barrier. Other processes that can reach the barrier but aren't yet at it continue executing. One-by-one, processes reach the barrier and, when they do, TotalView holds them.

When a process is *held*, it ignores commands that tell it to execute. This means, for example, that you can't tell it to go or to step. If, for some reason, you want the process to execute, you can manually release it using either the **Group > Release** or **Process > Release Threads** command.

When all processes that share a barrier reach it, TotalView changes their state from *held* to *released*, which means that they will no longer ignore a command that tells it to begin executing.

The following figure shows seven processes that are sharing the same barrier. (Processes that aren't affected by the barrier aren't shown.)

- First Block: All seven are running freely.
- Second Block: One process hits the barrier and is held. Six are executing.
- Third Block: Five of the processes have now hit the barrier and are being held. Two are executing.
- Fourth Block: All have hit the barrier. Because TotalView isn't waiting for anything else to reach the barrier, it changes the processes' state to *released*. Although they are released, none are executing.



Tip-of-the-Week

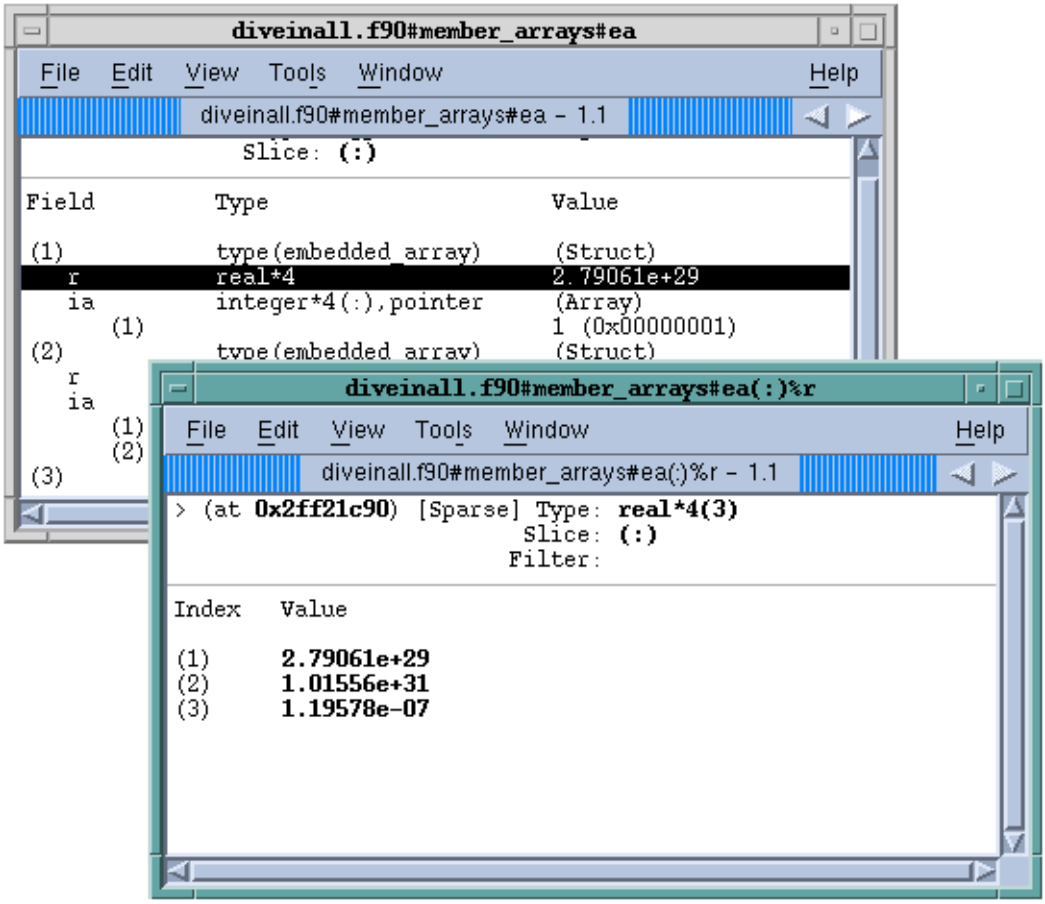
How do I display all values of one member in an array of structures?

Suppose you have the following Fortran definition:

```
type embedded_array
    real r
    integer, pointer :: ia(:)
end type embedded_array

type(embedded_array) ea (3)
```

After displaying **ea** in a Variable Window, select an **r** element, and then invoke the **View > Dive In All** command (which is also available when you right-click on a field). TotalView will respond by replacing the contents of the Variable Window with the three **r** elements of the **ea** array. These elements are treated as if they belong to a single array.



You can also use this command to unify the display of elements within a C array of structures as arrays. Here's a link to the page in the user guide that shows the results of diving on the **a** and the **next** elements in an array of structures.

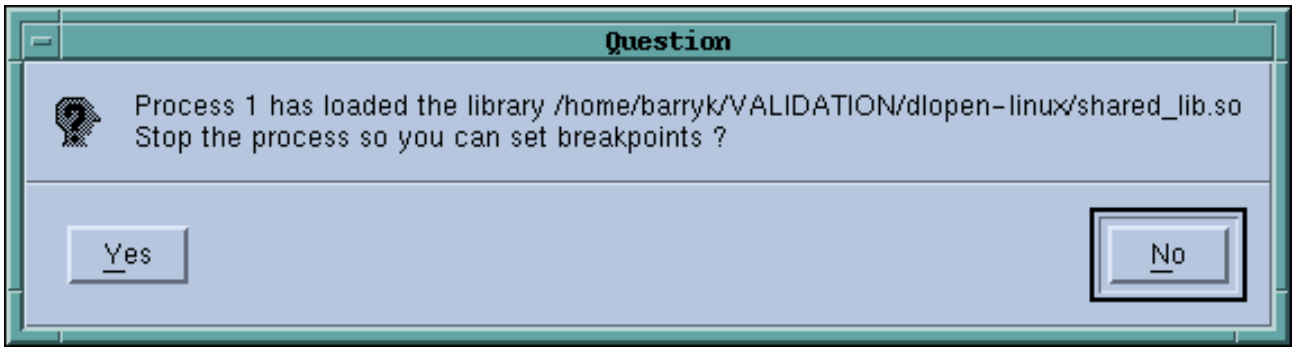
Tip-of-the-Week

How do I stop TotalView from popping up `*Stop Process*' questions?

This tip is based on a question submitted by
Ryan Olson from ESRI

When your program loads new processes, TotalView pops up question boxes asking if it should stop execution so that you can set breakpoints within the process. You could, of course, do other things.

TotalView asks these questions when your program uses a dynamically loaded library or when it starts other processes. Here's an example of one kind of question box:



Depending upon the number of times this happens, this can be extremely annoying.

You can stop these questions from appearing by selecting the **File > Preferences** command and making changes in one or both of the following pages:

- Clear the **Ask to stop when loading dynamic libraries** on the **Dynamic Libraries** page.
- Select the **Run the group** within the **When the job goes parallel or calls exec()** area on the **Parallel** page.

You can find tips that we've already sent out in our **Tip Archive**

Help us improve these tips!

- Tell us if you liked or disliked this tip.
- Is there something you'd like to be a tip?
- Tell us your tips.

Etnus, LLC
www.etnus.com
Phone: (508) 652-7700
documentation@etnus.com

Tip-of-the-Week

Why do some of my processes get stuck on a barrier?

In almost all cases, you've placed a barrier on a line that is only conditionally executed. Here's a simple conditional:

```
if (i_am_cold) {
    turn_on_heat();
} else {
    open_windows():
}
```

If you place a barrier on the **turn_on_heat()** function, the processes that take the other branch of the **if** statement will never hit the barrier. Because the barrier will never be satisfied-"satisfied" means that *all* of the processes have hit the barrier-TotalView will never release the processes held at **turn_on_heat()**.

Placing a barrier at both function calls doesn't help. Eventually, TotalView will be holding some processes at one place and the rest at the other. While everything is held, neither barrier is satisfied. This is because each needs the processes at the *other* barrier to reach it before TotalView will release that barrier's held processes. This means you can click on *step* and *go* commands until your mouse breaks and nothing will execute.

To get around this problem, create process-level breakpoints.

The moral of this tip is that you should only plant barriers in sections of code that are executed by all processes in the satisfaction set.

You can find tips that we've already sent out in our **Tip Archive**

Help us improve these tips!

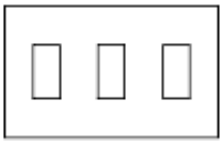
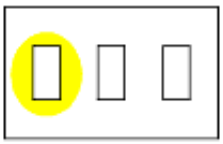
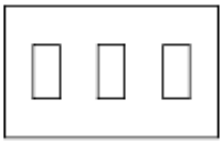
- Tell us if you liked or disliked this tip.
- Is there something you'd like to be a tip?
- Tell us your tips.

Etnus, LLC
www.etnus.com
Phone: (508) 652-7700
documentation@etnus.com

Tip-of-the-Week

Why are there two hold commands on the Process menu (part 2)?

Last week's tip explained why there are two hold commands. This week's continues the discussion by giving some examples.

Held/Release State	What Can Be Run Using Process > Go
	This figure shows a process with three threads. Before you do anything, all threads within the process can be run.
	Select the **Process > Hold** toggle. The button will be depressed. The blue indicates that you've hold the process. Nothing will run when you select **Process > Go**.
	Go to the **Threads** menu. Notice that the button next to the **Hold** command isn't selected. This is because the *thread hold* state is independent from the *process hold* state. Select it. The circle indicate that thread 1 is held. At this time, there are two different holds on thread 1. One is at process level, the other is at thread level. Nothing will run when you select **Process > Go**.
	Go back to the Process menu and reselect the **Hold** command. After you select Process > Go, the 2nd and 3rd threads run.
	Select **Process > Release Threads**. This releases the hold placed on the first thread by the **Thread > Hold** command. After you select Process > Go, all threads run.

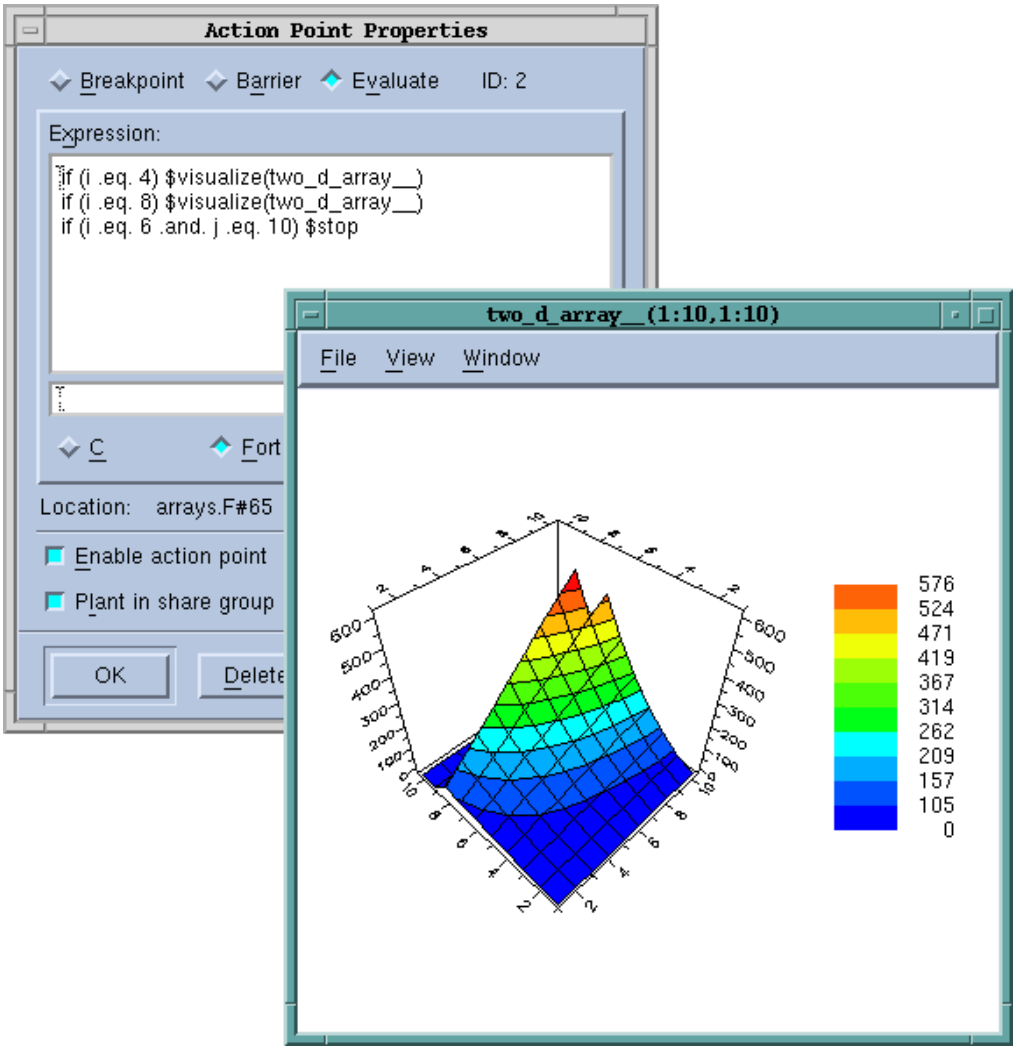
Tip-of-the-Week

Can TotalView animate how my array data is changing?

When you use the Variable Window's **Tools > Visualize** command, TotalView will bring up a new window containing a visual depiction of an array. The graph being displayed only changes when you again select the **Visualize** command.

In many cases, you would like TotalView to automatically update the graph so that you see how your data changes as your program executes. You can do this using an eval point. Here's how:

1. Select a line in your program and right click on the line number. Select **Properties** from the context menu. TotalView then displays its **Action Point Properties** Window.
2. Select the **Evaluate** Button at the top of the window and then use the **$visualize** function to name an array. Here's an example:



3. Click **OK**, then start your program by selecting the **Go** Button.

Drawing the graph can take a lot of time, which means that you really don't don't want to show every change that occurs. In this example, TotalView was told only to display the graph when **i** was either equal to 4 or 8. It was also told to halt when **i** was equal to 6. This was done so that I could stop execution to take a longer look at the graph.