

# GETTING STARTED WITH REPLAY ENGINE



VERSION 1.5

Copyright © 2009 by TotalView Technologies. All rights reserved.

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.

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:

[ftp://ftp.totalviewtech.com/support/toolworks/Microline\\_totalview.tar.Z](ftp://ftp.totalviewtech.com/support/toolworks/Microline_totalview.tar.Z).

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

# Contents



## Understanding ReplayEngine

ReplayEngine Overview .....	2
System Resources ReplayEngine Uses .....	2
Replaying Your Program .....	3
Threads and Processes .....	4
Attaching to Running Programs .....	4
Using ReplayEngine .....	4
CLI Changes .....	6

## Contents

# Understanding ReplayEngine



## TotalView Technologies ReplayEngine: A New Paradigm in Debugging

The hardest step in locating software bugs centers on working backward from a failure to the error that caused it. Conventional debugging techniques do not make it easy to find the cause of an error as they allow you to control program execution only in the forward direction.

Instead of going back to the beginning to try to recreate the conditions of a problem, ReplayEngine lets you start from the point of failure and work backward in time to find the cause. Recreating the conditions of a crash, sometimes the hardest problem in conventional forward debugging, is no longer necessary. You can now move to locate errors that occurred long before the failure they caused.

ReplayEngine is embedded within TotalView, which means you must know how to use TotalView. TotalView documentation is available , on our web site at <http://www.totalviewtech.com>.

---

## ReplayEngine Overview

---

ReplayEngine lets you move backward in your program. To do this, it saves state information as your program executes. This information includes the order in which your program executes and changes to its data. When ReplayEngine is saving state information, it is in its *record mode*.

The saved state information is the program's execution *history*.

Using a ReplayEngine command shifts ReplayEngine into its *replay mode*. In this mode, you can move to any previously executed statement. When you move to one of these statements, ReplayEngine displays its saved state information. The information you see in replay mode is identical to the information that you saw in record mode.

Most debugging commands work the same in replay mode as they do in record mode. Commands such as diving on a variable or setting a break-point work as you would expect them to. The debugging commands that do not work are those that change or alter a recorded state. Typically, these are commands that:

- Change a variable's value.
- Call functions that alter memory.
- Run threads asynchronously.

If your program calls a routine that displays information, the routine will not display this information. For example, suppose your program calls `printf()`. When the `printf()` is executed in record mode, it writes text. However, when the `printf()` is replayed, this text is not rewritten. Similarly, if your program unlinks a file in record mode; the file will not be linked before the unlink statement when you are in replay mode.

When executing in record mode, your program runs slower than it would run if you were not using ReplayEngine. Usually, you will not notice the extra execution time. However, when you are in replay mode, the computational overhead required to recreate the program's state may be noticeable. When it needs extra time, ReplayEngine displays a dialog box that allows you to cancel the operation.

### System Resources ReplayEngine Uses

---

ReplayEngine writes internal information in `/tmp`. Normally, very little space is used for this, but there are some situations where it can grow large, and if your system has a small `/tmp` area, ReplayEngine may fill it up. If this occurs, you can:

- Increase the amount of storage allocated to `/tmp`.
- Use the `TMPDIR` environment variable to point to another disk location.

- Define a special TotalView variable, `TVD_REPLAY_TMPDIR`, for ReplayEngine to use as the base directory for writing its temporary information. For example:

```
setenv TVD_REPLAY_TMPDIR /home/user/smith/replayTempDir
```

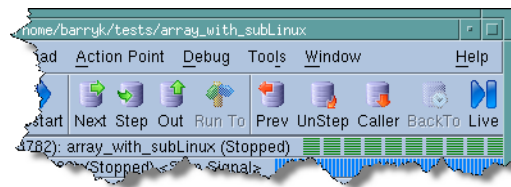
ReplayEngine also changes the amount of memory your program uses as it keeps history and state information in memory. For information on controlling the history information storage, see "Setting Preferences" on page 6.

While in replay mode. ReplayEngine creates extra processes, usually around ten, but you may see up to thirty. You should ignore these processes as they are only used by ReplayEngine.

## Replaying Your Program

Before you replay your program's statements, you must stop your program's execution. You can do this by halting your program or TotalView can stop execution when your program encounters a breakpoint. When execution stops, TotalView ungrays the ReplayEngine buttons you can use on its tool bar. (See Figure 1.)

Figure 1: ReplayEngine Tool Bar Commands



The ReplayEngine commands are as follows:

- **Prev**, which tells ReplayEngine to display the state that existed when the previous statement executed. If that line had a function call, **Prev** skips over the call.
- **Unstep**, which tells ReplayEngine to display the state that existed when the previous statement executed. If that line had a function call, ReplayEngine moves to the last statement in that function.
- **Caller**, which tells ReplayEngine to display the state that existed before the current routine was called.
- **BackTo**, which tells ReplayEngine to display the program's state for the line you select. This line must have executed prior to the currently displayed line. If you wish to move forward within replay mode, select a line and select the **Run To** button.
- **Live**, which tells ReplayEngine to shift from replay mode to record mode. It also displays the statement that would have executed if you had not moved into ReplayMode.



*The ReplayEngine tool bar commands only appear if you are using TotalView on a Linux-x86 or Linux-x86-64 machine. On these platforms, these buttons are permanently grayed out if you do not have a ReplayEngine license.*

When you need to move forward within the program's history, you can use the **Step**, **Next**, **Run To**, and **Out** buttons. These commands do the same thing in replay or record modes.

You can also set breakpoints in previously executed statements. After setting a breakpoint, pressing the **Go** button will move you to that statement. You can transform a breakpoint to an eval point if the eval point uses simple expressions such as `"if (x==y+z) $stop"`. You cannot, however, create barrier points.

If you reach the line that would have been executed if you hadn't gone into replay mode, you are automatically switched back to record mode and you can then resume program execution. You can also switch back to record mode by pressing the **Live** button.

### Threads and Processes

ReplayEngine runs one thread at a time, and it decides which thread will run in a multi-threaded or multi-process program. In record mode, ReplayEngine saves state information for each thread as it executes.

The order in which threads originally execute cannot be changed when you are in replay mode. In replay mode, all actions that occur must be in the same order as previously occurred.

If you need to control the way threads execute, use the TotalView asynchronous threading commands while in record mode. Using these commands you can:

- Single-step a process or lockstep group.
- Hold threads so they do not run.

If you are in replay mode, you cannot hold a thread or a process as they run in the same order as they ran originally.

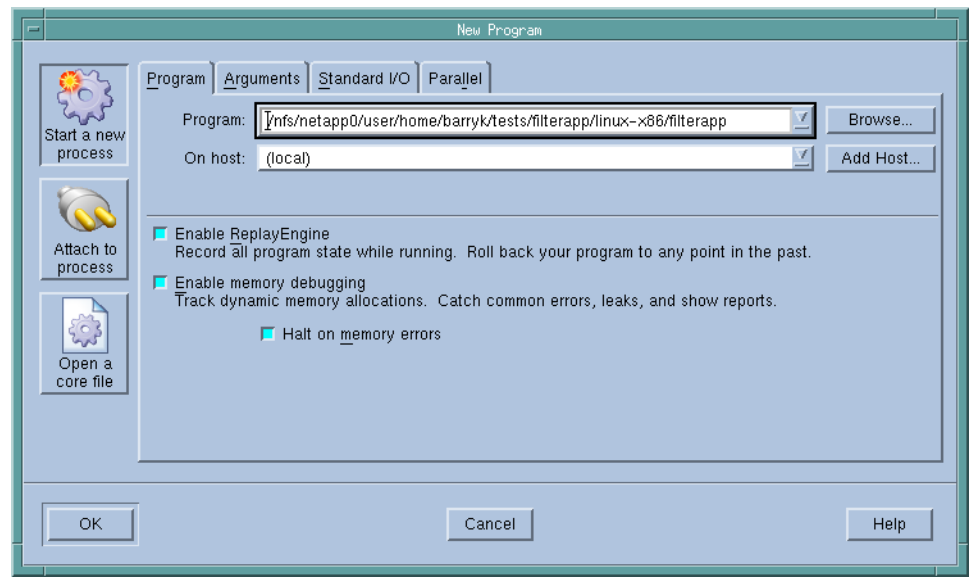
### Attaching to Running Programs

If you attach to a program, ReplayEngine begins recording that program's execution at the time you attached to it. This means that you cannot go back further than when you attached to it.

## Using ReplayEngine

There is very little difference between running TotalView and running ReplayEngine. The first step is enabling ReplayEngine. Do this by selecting **Enable Replay Engine** in the **File > New Program** dialog box or in the **Process > Startup Parameters** dialog box. Figure 2 shows the **New Program** dialog box.

Figure 2: Enabling using the File > New Command Dialog Box



If you are using the **New Program** dialog box, ReplayEngine begins recording instructions when you begin execution. If you are using the **Startup Parameters** dialog box, ReplayEngine is enabled when you restart your program.


You can also enable ReplayEngine by using the TotalView `-replay` command-line option.

After enabling ReplayEngine, you can begin controlling your program's execution using the same execution commands you use when ReplayEngine is not enabled. For example, you might set a breakpoint and press the **Go** button or select a line and press the **Run To** button.

When you wish to view the program's state, halt your program, then use the **Prev**, **UnStep**, **Caller**, or **BackTo** buttons to go to the statement you wish to examine. These four buttons are similar to the **Next**, **Step**, **Out**, and **Run To** tool bar buttons, differing only in that the Replay buttons go backwards in the program's history. The **Debug** pull-down menu contains the menu bar equivalents to these commands.

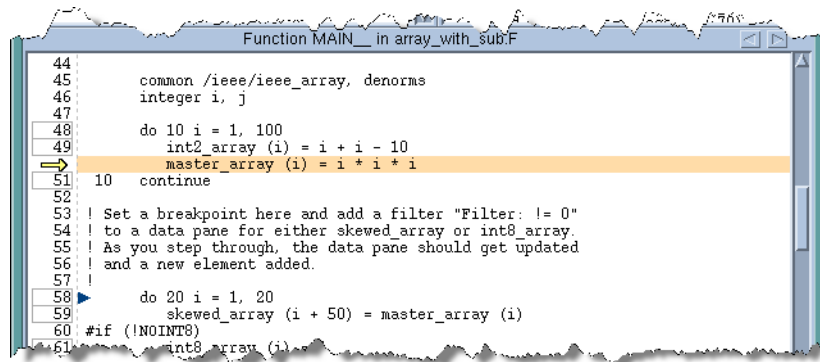
While you are in replay mode, notice that the **Next**, **Step**, **Out**, and **Run To** tool bar buttons are still displayed. This is because pressing these buttons moves you forward in the history.

When you're in replay mode, TotalView changes the highlight line from yellow to orange within the Source Pane. (See Figure 3.).

The Process window always shows the last line executed within record mode using the  symbol and the yellow highlight line is on the same line as this symbol. When you are in replay mode, this symbol is where ReplayEngine shifts from replay mode to record mode.

The scoping commands at the far left side of the tool bar have no effect in replay mode as the ReplayEngine only supports process width.

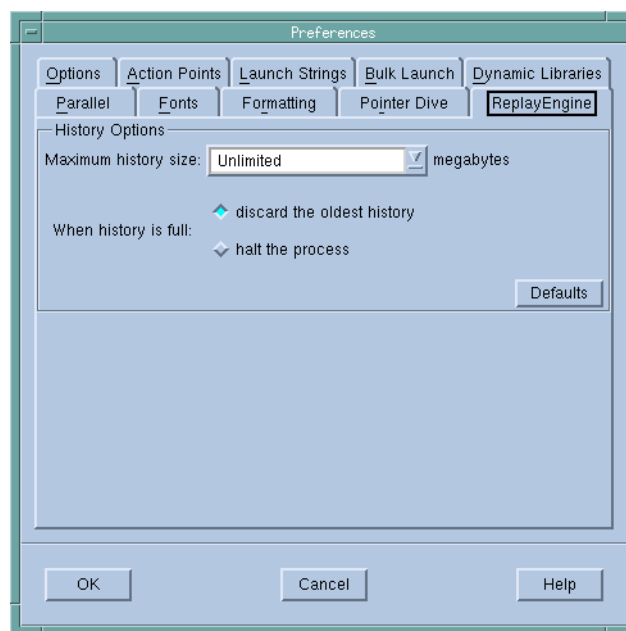
Figure 3: Source Pane While ReplayEngine is in Replay Mode



## Setting Preferences

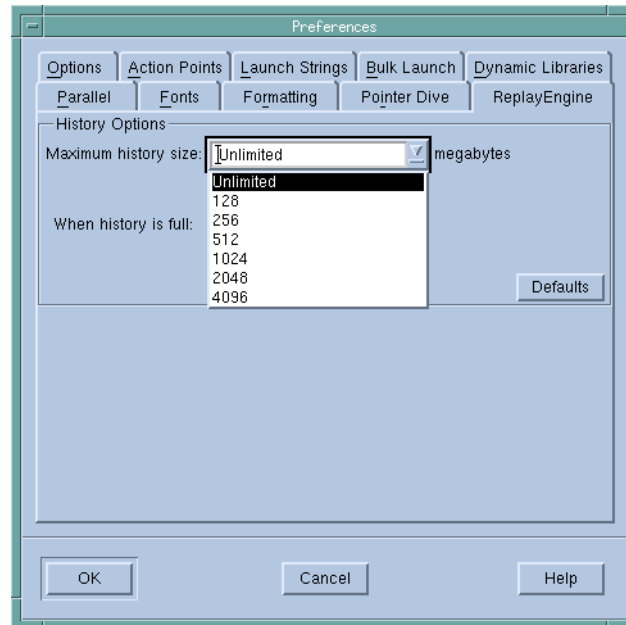
Use the **ReplayEngine** tab in the **Preferences** dialog box to define how ReplayEngine handles recorded history.

Figure 4: Preferences Dialog Box > ReplayEngine Page



The **Maximum history size** option sets the size in megabytes for ReplayEngine's history buffer. The default value, Unlimited, means ReplayEngine will use as much memory as is available to save recorded history. You can enter a new value into the text field or select from a drop-down list. (See Figure 5.)

Figure 5: Preferences Dialog Box > ReplayEngine Page Drop-Down



You can also set these options using the CLI as follows:

```
CLI: dset TV::replay_history_size <value>
```

For example:

```
dset TV::replay_history_size 1024M
```

sets the maximum history size to 1024 megabytes.

```
dset TV::replay_history_size 1000000
```

sets the maximum history size to 1000000 bytes.

The second option on the **ReplayEngine** preference page defines the tool's behavior when the history buffer is full. By default, the oldest history will be discarded so that recording can continue. You can change that so that the recording process will simply stop when the buffer is full.

You can also control this behavior using the CLI as follows:

```
CLI: dset TV::replay_history_mode <1,2>
```

For example:

```
dset TV::replay_history_mode 1
```

sets the mode to discard the oldest history and continue recording.

```
dset TV::replay_history_mode 2
```

sets the mode to stop the process when the buffer is full.

## CLI Support

- The **dload** and **dattach** CLI commands have the **-replay** option for enabling and disabling ReplayEngine. For example:  
`dload -replay myProgram`
- The **dnext**, **dnexti**, **dout**, **dstep**, **dstepi**, and **duntil** commands let you step backwards by using the **-back** option. For example:  
`dnext -back`  
`duntil -back 22`
- The **dhistory** command has the following options:
  - info** Dumps useful information about ReplayEngine.
  - get\_time** Displays the current time. The output of this command shows an integer value followed by an address. The first integer value is a virtual timestamp. This virtual timestamp does not refer to the exact point in time; it has a granularity that is typically a few lines of code. The address value is a PC value that corresponds to a precise point within that block of code.
  - go\_time *time*** Moves the process to an execution point represented by the *time* argument. The *time* argument is a virtual timestamp as reported by **dhistory -get\_time**. You cannot use this command to move to a specific instruction but you can use it to get to within a small block of code (usually within a few lines of your intended point in execution history). This command is typically used either for roughly bookmarking a point in a code or for searching execution history. It may need to be combined with stepping and **duntil** commands to return to an exact position.
  - go\_live** Resets the process back to record mode.
  - enable** Enables ReplayEngine for the next restart for the process.
  - disable** Disables Replay Engine for the next restart for the process.

These CLI commands are explained in detail in the *TotalView Reference Guide*.

# Index



## C

creating extra processes 3

## D

Debug pull-down menu 5

dload `-replay` and `-noreplay` options 6

dnext and dnexti `-back` command-line options 6

## E

enabling

in File > New Program dialog box 4

Process > Startup Parameters dialog box 4

using `-replay` command-line option 5

extra processes 3

## L

library and system calls 2

## P

ps command and extra processes 3

## R

record versus checkpoint 4

`-replay` command-line option 5

## S

seeing extra processes 3

switching to record mode 4

system and library calls 2

## T

tool bar buttons 5

## U

using ReplayEngine 4

