

MEMORYSCAPE

SCRIPTING



VERSION 2.4.0



Copyright © 2007–2008 by TotalView Technologies, LLC.
Copyright © 1999–2008 by Etnus LLC. All rights reserved.
Copyright © 1998–1999 by Etnus, Inc.
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 LLC. (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 LLC.

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

1 MemoryScape Scripting

display_specifiers Command-Line Option.....	1
event_action Command-Line Option	2
Other Command Line Options	2
memscript Example	3

MemoryScape Scripting

You can obtain information from MemoryScape by executing it in batch mode. Do this by invoking MemoryScape using the **memscript** command. Because MemoryScape will execute in batch mode, you must use command-line options to tell it what to do.

Here is what this command looks like:

```
memscript command_line_options
```

display_specifiers Command-Line Option

The **-display_specifiers** command-line option lets you control how MemoryScape writes information into the log file. Its format is as follows:

```
-display_specifiers "list_item"  
    Specifies one or more items that can be added or excluded from  
    the log file. Separate items with a comma.
```

list_item values are described in the following table. The word **no** in front of item suppresses the item's display.

Item	Controls display of ...
[no]show_backtrace	The backtrace for memory blocks
[no]show_backtrace_id	The backtrace ID for memory blocks
[no]show_block_address	The start and end addresses for a memory block
[no]show_flags	Memory block flags
[no]show_guard_details	Guard details for memory blocks
[no]show_guard_id	The guard ID for memory blocks
[no]show_guard_status	The guard status for memory blocks
[no]show_image	The process/library associated with a backtrace frame
[no]show_leak_stats	Leaked memory block statistics
[no]show_pc	The backtrace frame PC
[no]show_pid	The process PID

event_action Command-Line Option

The `-event_action` command-line option is the most complex of the command line option. Its format is as follows:

`-event_action "event=action list"`

Specifies one or more actions that the script should perform if an event occurs. The "event=action list" consists of comma-separated set *event=action* pairs. For example:

```
"alloc_null=save_memory_debugging_file, \
dealloc_notification=list_allocations"
```

event can be:

Event	Explanation
<code>addr_not_at_start</code>	A block is being freed, and the address is not at the beginning of the block.
<code>alloc_not_in_heap</code>	The block being freed is not in the heap.
<code>alloc_null</code>	The <code>malloc()</code> function returned a null block.
<code>alloc_returned_bad_alignment</code>	The block is misaligned.
<code>any_event</code>	All notification events.
<code>bad_alignment_argument</code>	The block returned by the <code>malloc</code> library is not aligned on a byte boundary required by your operating system. The heap may be corrupted. (This is not a program error.)
<code>double_dealloc</code>	Program is attempting to free a block already freed.
<code>free_not_allocated</code>	Program is attempting to free a block that was not allocated.
<code>guard_corruption</code>	Guard corruption was detected when program deallocated a block.

Event	Explanation
<code>realloc_not_allocated</code>	Program attempted to reallocate a block that was not allocated.
<code>termination_notification</code>	Program is about to execute its <code>_exit</code> routine.

action is as follows:

Action	Explanation
<code>check_guard_blocks</code>	Check for guard blocks and generates a corruption list.
<code>list_allocations</code>	Create a list of all your program's allocations.
<code>list_deallocations</code>	Create a list of all your program's deallocations.
<code>list_hoarded_blocks</code>	Create a list of all hoarded blocks.
<code>list_leaks</code>	Create a list of all of your program's leaks.
<code>save_html_heap_status_source_view</code>	Save the Heap Status Source report as an HTML file.
<code>save_memory_debugging_file</code>	Save a memory debugging file; you can reload this file at a later time.
<code>save_text_heap_status_source_view</code>	Save the Heap Status Source report as a text file.

Other Command Line Options

This section presents the other command line options that can use. The format of these options is much simpler than `-event_action`.

`-[no]guard_blocks`

Turn guard blocks on or off.

`-[no]hoard_freed_memory`

Turn the hoarding of freed memory on or off.

-[no]detect_leaks

Turn leak detection on or off.

-maxruntime *hh:mm:ss*

Specify the maximum amount of time the script should run where:

hh: number hours

mm: number of minutes

ss: number of seconds

As a script begins running, MemoryScape adds information to the beginning of the log file. This information includes time stamps for the file and for when processes start, the name of the program, and so on.

memscript Example

The following example performs the following actions:

- Runs the **filterapp** program under MemoryScape control.
- Passes an argument of **2** to the **filterapp** program.
- Whenever any event occurs—an HIA event, SEGV, and the like—saves a memory debugging file.
- Allows the script to run for no longer than 5 seconds.
- Perform the following activities: use guard blocks, hoard freed memory, and detect memory leaks.

```
memscript -maxruntime "00:00:05" \  
-event_action "any_event=save_memory_debugging_file" \  
-guard_blocks -hoard_freed_memory -detect_leaks \  
~/Work/filterapp -a 2
```


Index

- A**
addr_not_at_start event 2
alloc_not_in_heap event 2
alloc_null event 2
alloc_returned_bad_alignment event 2
any_event event 2
- B**
bad_alignment_argument event 2
- C**
check_guard_blocks action 2
- D**
-detect_leaks 3
-display_specifiers command-line option 1
double_dealloc event 2
- E**
-event_action command-line option 2
- F**
free_not_allocated event 2
- G**
-guard_blocks command-line option 2
guard_corruption event 2
- H**
-hoard_freed_memory command-line option 2
- L**
list_allocations action 2
list_deallocations action 2
list_hoarded_blocks action 2
list_leaks action 2
- M**
-maxruntime command-line option 3
- R**
realloc_not_allocated event 2
- S**
save_html_heap_status_source_view action 2
save_memory_debugging_file action 2
save_text_heap_status_source_view action 2
show_backtrace item 1
show_backtrace_id item 1
show_block_address item 1
show_flags item 1
show_guard_details item 1
show_guard_id item 1
show_guard_status item 1
show_image item 1
show_leak_stats item 1
show_pc item 1
show_pid item 1
- T**
termination_notification event 2

