

## **VMMMap**

Copyright © 2009-2010 Mark Russinovich and Bryce Cogswell

Sysinternals - [www.sysinternals.com](http://www.sysinternals.com)

Portions based on code by Jeffrey Richter

VMMMap is a process virtual and physical memory analysis utility. It shows a breakdown of a process's committed virtual memory types as well as the amount of physical memory (working set) assigned by the operating system to those types. Besides graphical representations of memory usage, VMMMap also shows summary information and a detailed process memory map. Powerful filtering, refresh and snapshot comparison capabilities allow you to identify the sources of process memory usage and the memory cost of application features.

Before reporting a bug, please make sure that you can reproduce the bug on the latest version of VMMMap posted at Sysinternals. To report a bug, email [markruss@microsoft.com](mailto:markruss@microsoft.com).

VMMMap works on Windows XP and higher, including x64 64-bit versions of Windows.

## Memory Types

VMMMap categorizes memory into one of several types:

### **Image**

The memory represents an executable file such as a .exe or .dll and has been loaded into a process by the image loader. It does not include images mapped as data files, which would be included in the Mapped File memory type. Image mappings can include shareable memory like code. When data regions, like initialized data, is modified, additional private memory is created in the process. The Details column shows the file's path.

### **Private**

Private memory is memory allocated by VirtualAlloc and not suballocated either by the Heap Manager or the .NET run time. It cannot be shared with other processes, is charged against the system commit limit, and typically contains application data.

### **Shareable**

Shareable memory is memory that can be shared with other processes, is backed by the paging file (if present), is charged against the system commit limit and typically contains data shared between DLLs in different processes or inter-process communication messages. The Windows APIs refer to this type of memory as pagefile-backed sections.

### **Mapped File**

The memory is shareable and represents a file on disk. The Details column shows the file's path. Mapped files typically contain application data.

### **Heap**

Heaps represent private memory managed by the user-mode heap manager and, like the Private memory type, is charged against the system commit limit and contains application data. Application memory allocations using the C runtime malloc library, HeapAlloc and LocalAlloc,

use Heap memory.

### **Managed Heap**

Managed heap represents private memory that's allocated and used by the .NET garbage collector and, like the Private memory type, is charged against the system commit limit and contains application data.

### **Stack**

Stacks are private memory used to store function parameters, local function variables and function invocation records for individual threads. Stacks are charged against the commit limit and typically grow on demand.

### **System**

System memory is private kernel-mode physical memory associated with the process. The vast majority of System memory consists of the process page tables.

### **Free**

Free memory regions are spaces in the process address space that are not allocated.

*Note: The VirtualProtect API can change the protections of any page to something different than that implied by the original allocation's memory type. That means that there can potentially be pages of memory private to the process in a shareable memory region, for instance, because the region was created as a pagefile-backed section, but then the application changed the protection on some pages to copy-on-write and modified them. The protection shown for a region isn't necessarily the protection it had since its creation.*

## The VMMap Window

When you run VMMap, it will present a process selection dialog. After you select a process it analyzes the process and presents the graphs:

- **Commit Summary Graph** This graph shows the committed (memory that represents data or code) memory usage of the process by type. The graph's scale is the total committed virtual memory usage of the process.
- **Private Summary Graph** This graph shows the committed private virtual memory. This memory is backed by the paging file and charged against the system commit limit. It corresponds to the PrivateBytes performance counter.
- **Working Set Summary Graph** This graph shows the working set usage of the process by memory type. Working set represents the amount of committed virtual memory that's in physical memory and owned by the process. The graph's scale is the total committed virtual memory.

The color key for the regions in the graphs is presented in the Summary View. Below the graphs VMMap shows two windows:

- **Summary View** This shows a summary of the virtual and physical usage of the process by type.
- **Details View** This shows the memory regions of the process address space.

For each region, VMMap displays the memory type, memory protection, and virtual and physical memory usage. Selecting a type in the Summary View filters the Details View to just show regions of the selected type. Select Total to show all memory types in the Details View. In order to reduce noise in the output, VMMap does not show entries that have a value of 0.

Both windows include the following columns of information:

**Size**

Total size of the allocated type or region. For the Summary View and regions in the Details View that do not have reserved areas, this is equal to the maximum amount of physical memory required to store the region's data.

**Committed**

The amount of the allocation backed by system virtual memory (RAM and paging files) and charged against the system commit limit.

**Private**

The amount of the allocation that, if modified, is private to the process (copy-on-write pages that have not been modified are included). This represents the charge to the system commit limit (sum of RAM plus the paging files) of the region.

**Total WS**

The amount of physical memory assigned to the type or region.

**Private WS**

The amount of physical memory assigned to the type or region that cannot be shared with other processes.

**Shareable WS**

The amount of physical memory assigned to the type or region that can be shared with other processes.

**Shared WS**

The amount of Shareable WS that is currently shared with other processes.

**Locked WS**

The amount of the working set that is locked into physical memory. This corresponds to memory locked via the VirtualAlloc API as well as Address Windowing Extensions (AWE) memory views. Note that working set figures returned by some other diagnostic tools does not

include AWE memory.

**Largest** The largest block of the particular size.

*Note: Because of limitations in the APIs provided by the operating system, on 64-bit Windows XP or 64-bit Windows Server 2003, Vmmap does not show the regions corresponding to 32-bit thread stacks when analyzing 32-bit processes.*

## Strings

In some cases, the purpose of a memory region can be revealed by the string data stored within it. To view printable strings (ASCII or UNICODE strings of three or more characters in length), select a region and then the Strings menu item from the Edit menu.



## Refreshing a Scan

You can refresh a scan by hitting F5 or selecting Refresh from the Refresh Menu. The Empty Working Set menu item in the Refresh menu releases all physical memory assigned to the process and then refreshes the scan. This feature is useful for measuring the memory cost of an application feature where you would empty the working set, exercise the feature and then refresh the display to look at how much physical memory the application referenced.

VMMMap saves every snapshot and you can view the history in the Timeline dialog. Select a particular snapshot by left-clicking in the graph area and moving the mouse to the desired snapshot point.

## Viewing Changes

VMMMap saves each snapshot and allows you to view the differences by selecting the Show Changes entry in the Options menu. When you toggle to that mode, the status bar shows the times of the refreshes being compared and the summary and details lists shows the differences between them. VMMMap shows address ranges that are in the most recent snapshot but not the previous one with a green highlight color and those that were deleted in red. You can toggle back to viewing the statistics of the most recent snapshot by deselecting the option.

You can view differences between any two snapshots by opening the Timeline dialog, clicking the mouse and dragging the selection region between the two points of interest.

## Tracing

You can have VMMap automatically take snapshots and capture a trace of several memory-related operations by opening the process selection dialog, switching to the Launch and Trace a New Program page, and entering the executable's path and command-line options. VMMap will generate snapshots at the interval configured in the Trace Snapshot Interval menu of the Options menu.

In addition, it will use Detours DLL injection library to instrument the heap and virtual allocation functions executed by the process. You can view the history of all recorded operations by opening the Trace history dialog using the Trace button on the main window. To view operations related to a particular heap region, select the heap block in the details pane and then click the Heap Allocations button. The Calltree button will show the call stacks of all places in the process from which the recorded memory allocations were made. Just like for manual snapshots, you can open the Timeline window to see the history of the process's execution, select particular snapshots for viewing, or select two snapshots to see the differences.

## Options

The options menu contains the following items:

- **Expand All** This expands all memory regions in the Details View
- **Collapse All** This collapses all memory regions in the Details View
- **Show Free Regions** Selecting this causes Details View to include free memory regions
- **Font** Use this to change the font of the Summary and Details Views.

## Saving, Loading and Copying

### Saving and Loading Scan Results

The Save menu item in the File menu includes several ways to save output from a VMMap scan. The Save dialog has options for the following output format:

- **.MMP** This is the native VMMap file format. Use this format if you want to load the output back into the VMMap display.
- **.CSV** This is comma-separated value output, which is ideal for generating output that you can easily import into Excel.
- **.TXT** This format is ideal for sharing the text form of scan results in a readable form.

Note that a saved .MMP file includes the two most recent snapshots, enabling you to view differences with the Refresh Shows Changes option when you load the file back into VMMap.

### Copying

The Edit menu includes two selections for copying output to the clipboard

- **Copy Address** This menu item copies just the address of the currently selected line in the Details View.
- **Copy All** This copies the text from the display, including the process name and ID, Summary View and Details View.

## Command Line Options

VMMMap supports the following command-line options:

**usage: vmmmap [-64] [-p <pid or process name> [outputfile]] [-o  
inputfile]**

- 64** Use the 64-bit version to analyze a 32-bit process instead of the 32-bit version.
- p** Process ID or process name. If you specify a name, VMMMap will match it against the first process that has a name that begins with the specified text.
- output file** If you specify an output file, VMMMap will scan the target process and then terminate. If you don't include an extension, VMMMap will add .mmp and save in its native format. Add a .csv extension to save as CSV format; any other extension will save as .txt.
- inputfile** Has VMMMap open the specified .mmp file on startup.