msdn.microsoft.com

SOS.dll (SOS Debugging Extension)

Command

Description

AnalyzeOOM (**ao**)

Displays the information for the last OOM that occurred on an allocation request to the garbage collection heap. (In server garbage collection, it displays OOM, if any, on each garbage collection heap.)

BPMD [**-nofuturemodule**] [<*module name*> <*method name*>] [**-md** <*MethodDesc*>] **-list -clear** <*pending breakpoint number*> **-clearall**

Creates a breakpoint at the specified method in the specified module.

If the specified module and method have not been loaded, this command waits for a notification that the module was loaded and just-in-time (JIT) compiled before creating a breakpoint.

You can manage the list of pending breakpoints by using the **-list**, **-clear**, and **-clearall** options:

- The **-list** option generates a list of all the pending breakpoints. If

a pending breakpoint has a non-zero module ID, that breakpoint is specific to a function in that particular loaded module. If the pending breakpoint has a zero module ID, that breakpoint applies to modules that have not yet been loaded.

- Use the **-clear** or **-clearall** option to remove pending breakpoints from the list.

CLRStack [**-a**] [**-l**] [**-p**] [**-n**]

Provides a stack trace of managed code only.

- The **-p** option shows arguments to the managed function.

- The **-l** option shows information on local variables in a frame. The SOS Debugging Extension cannot retrieve local names, so the output for local names is in the format <*local address*> **=** <*value*>.

- The **-a** (all) option is a shortcut for **-l** and **-p** combined.

- The **-n** option disables the display of source file names and line numbers. If the debugger has the option SYMOPT_LOAD_LINES specified, SOS will look up the symbols for every managed frame and if successful will display the corresponding source file name and line number. The **-n** (No line numbers) parameter can be specified to disable this behavior.

The SOS Debugging Extension does not display transition frames on x64 and IA-64-based platforms.

COMState

Lists the COM apartment model for each thread and a **Context** pointer, if available.

DumpArray [**-start** <*startIndex*>] [**-length** <*length*>] [**-details**] [**-nofields**] <*array object address*>

-or-

DA [**-start** <*startIndex*>] [**-length** <*length*>] [**-detail**] [**-nofields**] *array object address*>

Examines elements of an array object.

- The **-start** option specifies the starting index at which to display elements.
- The **-length** option specifies how many elements to show.
- The **-details** option displays details of the element using the **DumpObj** and **DumpVC** formats.
- The **-nofields** option prevents arrays from displaying. This option is available only when the **-detail** option is specified.

DumpAssembly <*assembly address*>

Displays information about an assembly.

The **DumpAssembly** command lists multiple modules, if they exist.

You can get an assembly address by using the **DumpDomain** command.

DumpClass <*EEClass address*>

Displays information about the **EEClass** structure associated with a type.

The **DumpClass** command displays static field values but does not display nonstatic field values.

Use the **DumpMT**, **DumpObj**, **Name2EE**, or **Token2EE** command to get an **EEClass** structure address.

DumpDomain [<*domain address*>]

Enumerates each Assembly object that is loaded within the specified AppDomain object address. When called with no parameters, the **DumpDomain** command lists all AppDomain objects in a process.

DumpHeap [**-stat**] [**-strings**] [**-short**] [**-min** <*size*>] [**-max** <*size*>] [**-thinlock**] [**-startAtLowerBound**] [**-mt** <*MethodTable address*>] [**-type** <*partial type name*>][*start* [*end*]]

Displays information about the garbage-collected heap and collection statistics about objects.

The **DumpHeap** command displays a warning if it detects excessive

fragmentation in the garbage collector heap.

- The **-stat** option restricts the output to the statistical type summary.

- The **-strings** option restricts the output to a statistical string value summary.

- The **-short** option limits output to just the address of each object. This lets you easily pipe output from the command to another debugger command for automation.

- The **-min** option ignores objects that are less than the *size* parameter, specified in bytes.

- The **-max** option ignores objects that are larger than the *size* parameter, specified in bytes.

- The **-thinlock** option reports ThinLocks. For more information, see the **SyncBlk** command.

- The **-startAtLowerBound** option forces the heap walk to begin at the lower bound of a supplied address range. During the planning phase, the heap is often not walkable because objects are being moved. This option forces **DumpHeap** to begin its walk at the specified lower bound. You must supply the address of a valid object as the lower bound for this option to work. You can display memory at the address of a bad object to manually find the next method table. If the garbage collection is currently in a call to **memcopy**, you may also be able to find the address of

the next object by adding the size to the start address, which is supplied as a parameter.

- The **-mt** option lists only those objects that correspond to the specified **MethodTable** structure.

- The **-type** option lists only those objects whose type name is a substring match of the specified string.

- The *start* parameter begins listing from the specified address.

- The *end* parameter stops listing at the specified address.

DumpIL <*Managed DynamicMethod object*> | <*DynamicMethodDesc pointer*> | <*MethodDesc pointer*>

Displays the Microsoft intermediate language (MSIL) that is associated with a managed method.

Note that dynamic MSIL is emitted differently than MSIL that is loaded from an assembly. Dynamic MSIL refers to objects in a managed object array rather than to metadata tokens.

DumpLog [**-addr** <*addressOfStressLog*>] [<*Filename*>]

Writes the contents of an in-memory stress log to the specified file. If you do not specify a name, this command creates a file called StressLog.txt in the current directory.

The in-memory stress log helps you diagnose stress failures without

using locks or I/O. To enable the stress log, set the following registry keys under HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft \.NETFramework:

(DWORD) StressLog = 1

(DWORD) LogFacility = 0xffffffff

(DWORD) StressLogSize = 65536

The optional **-addr** option lets you specify a stress log other than the default log.

DumpMD <*MethodDesc address*>

Displays information about a **MethodDesc** structure at the specified address.

You can use the **IP2MD** command to get the **MethodDesc** structure address from a managed function.

DumpMT [**-MD**] <*MethodTable address*>

Displays information about a method table at the specified address. Specifying the **-MD** option displays a list of all methods defined with the object.

Each managed object contains a method table pointer.

DumpMethodSig <*sigaddr*> <*moduleaddr*>

Displays information about a **MethodSig** structure at the specified address.

DumpModule [**-mt**] <*Module address*>

Displays information about a module at the specified address. The **-mt** option displays the types defined in a module and the types referenced by the module

You can use the **DumpDomain** or **DumpAssembly** command to retrieve a module's address.

DumpObj [**-nofields**] <*object address*>

-or-

DO <*object address*>

Displays information about an object at the specified address. The **DumpObj** command displays the fields, the **EEClass** structure information, the method table, and the size of the object.

You can use the **DumpStackObjects** command to retrieve an object's address.

Note that you can run the **DumpObj** command on fields of type **CLASS** because they are also objects.

The **-nofields** option prevents fields of the object being displayed, it is useful for objects like String.

DumpRuntimeTypes

Displays the runtime type objects in the garbage collector heap and lists their associated type names and method tables.

DumpStack [**-EE**] [**-n**] [*top stack* [*bottom stack*]]

Displays a stack trace.

- The **-EE** option causes the **DumpStack** command to display only managed functions. Use the *top* and *bottom* parameters to limit the stack frames displayed on x86 platforms.

- The **-n** option disables the display of source file names and line numbers. If the debugger has the option SYMOPT_LOAD_LINES specified, SOS will look up the symbols for every managed frame and if successful will display the corresponding source file name and line number. The **-n** (No line numbers) parameter can be specified to disable this behavior.

On x86 and x64 platforms, the **DumpStack** command creates a verbose stack trace.

On IA-64-based platforms, the **DumpStack** command mimics the debugger's **K** command. The *top* and *bottom* parameters are ignored on IA-64-based platforms.

DumpSig <*sigaddr*> <*moduleaddr*>

Displays information about a **Sig** structure at the specified address.

DumpSigElem <*sigaddr*> <*moduleaddr*>

Displays a single element of a signature object. In most cases, you should use **DumpSig** to look at individual signature objects. However, if a signature has been corrupted in some way, you can use **DumpSigElem** to read the valid portions of it.

DumpStackObjects [**-verify**] [*top stack* [*bottom stack*]]

-or-

DSO [**-verify**] [*top stack* [*bottom stack*]]

Displays all managed objects found within the bounds of the current stack.

The **-verify** option validates each non-static **CLASS** field of an object field.

Use the **DumpStackObject** command with stack tracing commands such as the **K** command and the **CLRStack** command to determine the values of local variables and parameters.

DumpVC <*MethodTable address*> <*Address*>

Displays information about the fields of a value class at the specified address.

The **MethodTable** parameter allows the **DumpVC** command to correctly interpret fields. Value classes do not have a method table

as their first field.

EEHeap [**-gc**] [**-loader**]

Displays information about process memory consumed by internal common language runtime data structures.

The **-gc** and **-loader** options limit the output of this command to garbage collector or loader data structures.

The information for the garbage collector lists the ranges of each segment in the managed heap. If the pointer falls within a segment range given by **-gc**, the pointer is an object pointer.

EEStack [**-short**] [**-EE**]

Runs the **DumpStack** command on all threads in the process.

The **-EE** option is passed directly to the **DumpStack** command. The **-short** parameter limits the output to the following kinds of threads:

- Threads that have taken a lock.
- Threads that have been stalled in order to allow a garbage collection.
- Threads that are currently in managed code.

EEVersion

Displays the common language runtime version.

EHInfo [<*MethodDesc address*>] [<*Code address*>]

Displays the exception handling blocks in a specified method. This command displays the code addresses and offsets for the clause block (the **try** block) and the handler block (the **catch** block).

FAQ

Displays frequently asked questions.

FinalizeQueue [**-detail**] | [**-allReady**] [**-short**]

Displays all objects registered for finalization.

- The **-detail** option displays extra information about any **SyncBlocks** that need to be cleaned up, and any **RuntimeCallableWrappers** (RCWs) that await cleanup. Both of these data structures are cached and cleaned up by the finalizer thread when it runs.

- The **-allReady** option displays all objects that are ready for finalization, regardless of whether they are already marked by the garbage collection as such, or will be marked by the next garbage collection. The objects that are in the "ready for finalization" list are finalizable objects that are no longer rooted. This option can be very expensive, because it verifies whether all the objects in the finalizable queues are still rooted.

- The **-short** option limits the output to the address of each object. If it is used in conjunction with **-allReady**, it enumerates all objects that have a finalizer that are no longer rooted. If it is used independently, it lists all objects in the finalizable and "ready for finalization" queues.

-

FindAppDomain <*Object address*>

Determines the application domain of an object at the specified address.

FindRoots -gen <*N*> | **-gen any** | <*object address*>

Causes the debugger to break in the debuggee on the next collection of the specified generation. The effect is reset as soon as the break occurs. To break on the next collection, you have to reissue the command. The <*object address*> form of this command is used after the break caused by the **-gen** or **-gen any** has occurred. At that time, the debuggee is in the right state for **FindRoots** to identify roots for objects from the current condemned generations.

GCHandles [**-perdomain**]

Displays statistics about garbage collector handles in the process.

The **-perdomain** option arranges the statistics by application domain.

Use the **GCHandles** command to find memory leaks caused by garbage collector handle leaks. For example, a memory leak occurs when code retains a large array because a strong garbage collector handle still points to it, and the handle is discarded without freeing it.

GCHandleLeaks

Searches memory for any references to strong and pinned garbage collector handles in the process and displays the results. If a handle is found, the **GCHandleLeaks** command displays the address of the reference. If a handle is not found in memory, this command displays a notification.

GCInfo <*MethodDesc address*> <*Code address*>

Displays data that indicates when registers or stack locations contain managed objects. If a garbage collection occurs, the collector must know the locations of references to objects so it can update them with new object pointer values.

GCRoot [**-nostacks**] <*Object address*>

Displays information about references (or roots) to an object at the specified address.

The **GCRoot** command examines the entire managed heap and the handle table for handles within other objects and handles on the stack. Each stack is then searched for pointers to objects, and the finalizer queue is also searched.

This command does not determine whether a stack root is valid or is discarded. Use the **CLRStack** and **U** commands to disassemble the frame that the local or argument value belongs to in order to determine if the stack root is still in use.

The **-nostacks** option restricts the search to garbage collector handles and freachable objects.

GCWhere <*object address*>

Displays the location and size in the garbage collection heap of the argument passed in. When the argument lies in the managed heap but is not a valid object address, the size is displayed as 0 (zero).

help [<*command*>] [*faq*]

Displays all available commands when no parameter is specified, or displays detailed help information about the specified command.

The *faq* parameter displays answers to frequently asked questions.

HeapStat [**-inclUnrooted** | **-iu**]

Displays the generation sizes for each heap and the total free space in each generation on each heap. If the -**inclUnrooted** option is specified, the report includes information about the managed objects from the garbage collection heap that is no longer rooted.

HistClear

Releases any resources used by the family of **Hist** commands.

Generally, you do not have to explicitly call **HistClear**, because each **HistInit** cleans up the previous resources.

HistInit

Initializes the SOS structures from the stress log saved in the debuggee.

HistObj <*obj_address*>

Examines all stress log relocation records and displays the chain of garbage collection relocations that may have led to the address passed in as an argument.

HisttObjFind <*obj_address*>

Displays all the log entries that reference an object at the specified address.

HistRoot <*root*>

Displays information related to both promotions and relocations of the specified root.

The root value can be used to track the movement of an object through the garbage collections.

IP2MD <*Code address*>

Displays the **MethodDesc** structure at the specified address in code that has been JIT-compiled.

ListNearObj (**lno**) <*obj_address*>

Displays the objects preceding and following the specified address. The command looks for the address in the garbage collection heap that looks like a valid beginning of a managed object (based on a valid method table) and the object following the argument address.

MinidumpMode [**0**] [**1**]

Prevents running unsafe commands when using a minidump.

Pass **0** to disable this feature or **1** to enable this feature. By default, the **MinidumpMode** value is set to **0**.

Minidumps created with the **.dump /m** command or **.dump** command have limited CLR-specific data and allow you to run only a subset of SOS commands correctly. Some commands may fail with unexpected errors because required areas of memory are not mapped or are only partially mapped. This option protects you from running unsafe commands against minidumps.

Name2EE <*module name*> <*type or method name*>

-or-

Name2EE <*module name*>**!**<*type or method name*>

Displays the **MethodTable** structure and **EEClass** structure for the specified type or method in the specified module.

The specified module must be loaded in the process.

To get the proper type name, browse the module by using the Ildasm.exe (IL Disassembler). You can also pass * as the module name parameter to search all loaded managed modules. The *module name* parameter can also be the debugger's name for a module, such as mscorlib or image00400000.

This command supports the Windows debugger syntax of <*module*>**!**<*type*>. The type must be fully qualified.

ObjSize [<*Object address*>] | [**-aggregate**] [**-stat**]

Displays the size of the specified object. If you do not specify any parameters, the **ObjSize** command displays the size of all objects found on managed threads, displays all garbage collector handles in the process, and totals the size of any objects pointed to by those handles. The **ObjSize** command includes the size of all child objects in addition to the parent.

The **-aggregate** option can be used in conjunction with the **-stat** argument to get a detailed view of the types that are still rooted. By using **!dumpheap -stat** and **!objsize -aggregate -stat**, you can determine which objects are no longer rooted and diagnose various memory issues.

PrintException [**-nested**] [**-lines**] [<*Exception object address*>]

-or-

PE [**-nested**] [<*Exception object address*>]

Displays and formats fields of any object derived from the Exception class at the specified address. If you do not specify an address, the **PrintException** command displays the last exception thrown on the current thread.

- The **-nested** option displays details about nested exception objects.

- The **-lines** option displays source information, if available.

You can use this command to format and view the **_stackTrace** field, which is a binary array.

ProcInfo [**-env**] [**-time**] [**-mem**]

Displays environment variables for the process, kernel CPU time, and memory usage statistics.

RCWCleanupList <*RCWCleanupList address*>

Displays the list of runtime callable wrappers at the specified address that are awaiting cleanup.

SaveModule <*Base address*> <*Filename*>

Writes an image, which is loaded in memory at the specified

address, to the specified file.

SOSFlush

Flushes an internal SOS cache.

StopOnException [**-derived**] [**-create** | **-create2**] <*Exception*> <*Pseudo-register number*>

Causes the debugger to stop when the specified exception is thrown, but to continue running when other exceptions are thrown.

The **-derived** option catches the specified exception and every exception that derives from the specified exception.

SyncBlk [**-all** | <*syncblk number*>]

Displays the specified **SyncBlock** structure or all **SyncBlock** structures. If you do not pass any arguments, the **SyncBlk** command displays the **SyncBlock** structure corresponding to objects that are owned by a thread.

A **SyncBlock** structure is a container for extra information that does not need to be created for every object. It can hold COM interop data, hash codes, and locking information for thread-safe operations.

ThreadPool

Displays information about the managed thread pool, including the

number of work requests in the queue, the number of completion port threads, and the number of timers.

Token2EE <*module name*> <*token*>

Turns the specified metadata token in the specified module into a **MethodTable** structure or **MethodDesc** structure.

You can pass * for the module name parameter to find what that token maps to in every loaded managed module. You can also pass the debugger's name for a module, such as mscorlib or image00400000.

Threads [**-live**] [**-special**]

Displays all managed threads in the process.

The **Threads** command displays the debugger shorthand ID, the common language runtime thread ID, and the operating system thread ID. Additionally, the **Threads** command displays a Domain column that indicates the application domain in which a thread is executing, an APT column that displays the COM apartment mode, and an Exception column that displays the last exception thrown in the thread.

- The **-live** option displays threads associated with a live thread.

- The **-special** option displays all special threads created by the CLR. Special threads include garbage collection threads (in concurrent and server garbage collection), debugger helper

threads, finalizer threads, AppDomain unload threads, and thread pool timer threads.

ThreadState <*State value field***>**

Displays the state of the thread. The *value* parameter is the value of the **State** field in the **Threads** report output.

Example:

Copy 0:003> !Threads ThreadCount: 2 UnstartedThread: 0 BackgroundThread: 1 PendingThread: 0 DeadThread: 0 Hosted Runtime: no PreEmptive GC Alloc Lock ID OSID ThreadOBJ State GC Context Domain Count APT Exception 0 1 250 0019b068 a020 Disabled 02349668:02349fe8 0015def0 0 MTA 2 2 944 001a6020 b220 Enabled 00000000:00000000 0015def0 0 MTA (Finalizer) 0:003> !ThreadState b220 Legal to Join Background CLR Owns CoInitialized In Multi Threaded Apartment

TraverseHeap [**-xml**] <*filename*>

Writes heap information to the specified file in a format understood by the CLR profiler. The **-xml** option causes the **TraverseHeap** command to format the file as XML.

You can download the CLR Profiler from the Microsoft Download Center.

U [**-gcinfo**] [**-ehinfo**] [**-n**] <*MethodDesc address*> | <*Code address*>

Displays an annotated disassembly of a managed method specified

either by a **MethodDesc** structure pointer for the method or by a code address within the method body. The **U** command displays the entire method from start to finish, with annotations that convert metadata tokens to names.

- The **-gcinfo** option causes the **U** command to display the **GCInfo** structure for the method.

- The **-ehinfo** option displays exception information for the method. You can also obtain this information with the **EHInfo** command.

- The **-n** option disables the display of source file names and line numbers. If the debugger has the option SYMOPT_LOAD_LINES specified, SOS looks up the symbols for every managed frame and, if successful, displays the corresponding source file name and line number. You can specify the **-n** option to disable this behavior.

VerifyHeap

Checks the garbage collector heap for signs of corruption and displays any errors found.

Heap corruptions can be caused by platform invoke calls that are constructed incorrectly.

VerifyObj <*object address*>

Checks the object that is passed as an argument for signs of

corruption.

VMMap

Traverses the virtual address space and displays the type of protection applied to each region.

VMStat

Provides a summary view of the virtual address space, ordered by each type of protection applied to that memory (free, reserved, committed, private, mapped, image). The TOTAL column displays the result of the AVERAGE column multiplied by the BLK COUNT column.