WinDbg Cheat Sheet - Data Structures, Commands and Extensions

Date: 22/06/2020
Author: Harry Miller (@MOV_EDX)
Website: https://bsodtutorials.wordpress.com

Contents

Processes and Threads:

!process [/s Session] [/m Module] [Process [Flags]]

Displays the _EPROCESS structure for a given process. To list all the active processes, then use ! process 0 0. If you wish to view the _KPROCESS structure, then please use the dt command.

Flags:

Flag Value	Flag Description
0	Displays time and priority statistics for the process
1	Displays the associated threads and their events and wait states.

2	Displays threads and their call stacks.
3	Display the return address and the stack pointer for each thread which is associated to the process.

!processfields (Windows 2000 only)

Displays the members of the _EPROCESS structure along with their offsets. This command has now been replaced with **dt _EPROCESS**.

!processirps [Process Address]

Displays all the IRPs which are associated to a process. If you provide a process address, then the IRPs associated to that particular process will be displayed.

Flags:

Flag Value	Flag Description
0	Displays IRPs queued to a thread.
1	Displays IRPs which are queued to file objects.

.tlist [Flags] [FileNamePattern] (Live Debugging only)

Displays all the processes which are currently running on the system.

Flags:

-v	Displays additional information about the processes including session information.
-c	Displays the current process only.

The second parameter is either the name of the process to be displayed or a pattern which a process name must match.

!peb [PEB Address]

Displays the process environment block for a given process. Please note that this command will only work with user-mode memory dumps and full memory dump files.

.process [Process Address]

Back to Contents

Switches the current process context to the one specified. If no process address or name is provided, then the process context will be set to the last running process.

Please note that the process address is technically the address of the _EPROCESS structure.

!token [Token Address]

Displays information about a specified access token.

Data Structures

_EPROCESS, _KPROCESS, _PEB, _TOKEN_TYPE, _TOKEN, _SID, _SID_NAME_USE, _TOKEN_SOURCE, _SEP_TOKEN_PRIVILEGES, _SECURITY_IMPERSONATION_LEVEL, _TOKEN_CONTROL, _SID_AND_ATTRIBUTES_HASH

Global Variables

PsActiveProcessHead – head of a doubly linked list of all active processes on a given processor.
PsInitialSystemProcess – the starting process (System) and the first process within the linked list.

!thread [-p] [-t] [Thread Address [Flags]]

Displays the _ETHREAD structure for a given thread. If you wish to view the _KTHREAD structure then use the dt command.

Flags:

Flag Value	Flag Description
1	Displays the thread wait states
2	Displays the call stack for the thread; this flag must be used with the first flag.
3	Displays the return address and stack pointer for the thread's call stack.
4	Changes the process context to that of the process which the thread is associated to.
-p	Includes summary information regarding the associated process for the specified thread.
-t	Enables you to pass a thread ID (TID) rather than an address to the command.

!threadfields (Windows 2000 only)

Displays the members of the _ETHREAD structure along with their offsets. This command has now

been replaced with **dt _ETHREAD**.

!tp [Address Type] Address [Flags]]

Displays thread pool information; thread pools are used to manage worker threads on behalf of a process.

Address Types:

pool – This will display the entire thread pool at the given address.

tqueue – This will display all the active timers for the given address.

Flags:

Flag Value	**Flag Description**
1	Displays output as a single line
2	Display thread pool member information
3	

!teb [TEB Address]

Displays the thread environment block for a given thread. Please note that this command will only work with user-mode memory dumps and full memory dump files.

.thread [Thread Address]

Switches the current thread context to the one specified. If no thread address is provided, then the process context will be set to the last running thread.

Please note that the thread address is technically the address of the _ETHREAD structure.

Data Structures:

_ETHREAD, _KTHREAD and _TEB

Stack Traces and Stack Frames:

k

Displays the smallest amount of detail for a stack trace, includes the following: stack frame address, return address and the function name.

Back to Contents

knL

Similar to k, but includes the stack frame number in addition to the above.

kb

The stack frame number is omitted, however, the arguments are also provided for each stack frame.

kv

This command includes the greatest amount of information, along with what the other commands provide, it also includes any frame pointer omission optimisations along with trap frames which have been saved.

kF

Similar to knL, but includes the addition of the memory usage for each stack frame.

.kframes [FrameCount]

Sets the default number of frames which one of the **k** commands will display. By default, the stack frame count is set to 20.

.frame [/r] [Frame Number]

Displays the current stack frame, along with its registers if the /r switch has been provided to the command. This is very useful when you want to examine the parameters values which have been passed to a function.

!uniqstack [-b | -v | -p] [-n]

Displays all stacks for all threads within the current process.

Flags:

Flag Value	Flag Description
-b	Displays the first three parameters for each stack frame
-v	Displays Frame Pointer Omission (FPO) information if applicable
-p	Displays all the parameters for each stack frame along with their data types
-n	Displays the stack frame numbers

Back to Contents

!findstack [Symbol] [Display Level]

Displays stacks with a specified module or symbol.

Flags:

Flag Value	Flag Description
0	Displays only the TID for each thread
1	Displays both the TID and the stack frame for each thread
2	Displays the entire call stack for each thread.

!stacks [Detail [Filter]]

Displays all the call stacks in for each thread, a filter string can be provided which in turn will only display call stacks which contain the filter string as a substring in one of their symbols.

Flags:

Flag Value	Flag Description
0	Display summary information, this is the default value.
1	Displays summary information along with stacks which have been paged out.
2	Displays full verbose information for each kernel stack, including stacks which have been paged out.

Pools and Pool Allocations:

!frag [Flags]

This is no longer supported in newer versions of WinDbg.

!poolfind [Tag [Pool Type]

Searches paged and non-paged pool for pool allocations with the associated pool tag.

Pool Types:

Pool Value	Pool Type

0	Non-paged pool
1	Paged pool
2	Special Pool
4	Session Pool

!pooltag [Tag]

Displays the associated driver for a pool tag.

!poolused [Flags [Tag]]

Displays all the pool allocations for a specified pool tag. This includes paged and non-paged pool.

Flags:

Flag Value	**Flag Description**
0	Displays more verbose information about each allocation.
1	Sorts by non-paged pool usage.
2	Sorts by paged pool usage.
3	Displays session pool instead of paged and non-paged pool.

!pool [Address [Flags]]

Displays information about a specific pool allocation or pool page.

Flags:

Flag Value	**Flag Description**
0	Displays the pool contents and pool headers for the pool page.
1	Suppresses pool header information for all pool pages, apart from the pool allocation which matches the provided address
31	Suppresses pool type and pool tag information.

!poolval [Address [Display Level]]

Checks pool headers for consistency and any possible pool page corruption.

Back to Contents

Flags:

Flag Value	Flag Description
0	Displays basic information about the pool page
1	Includes linked header lists to be displayed too.
2	In addition to the above, includes pool header information.
3	Instead of basic header information, displays more verbose pool header information.

Data Structures

POOL_HEADER, POOL_TYPE and POOL_DESCRIPTOR, POOL_TRACKER_TABLE

Virtual and Physical Memory:

!vm [Flags]

Displays summary information about virtual memory

Flags:

Flag Value	Flag Description
0	Omits process specific information
1	Displays memory management stack information
2	Displays terminal server usage
3	Displays page file write log
4	Displays working set owner thread stacks
5	Displays kernel address space usage statistics

!vprot [Address]

Displays the virtual memory protection bits for a given address. For example, if a page is able to written to or not.

!vadump [-v]

Displays the virtual memory protection bits for all address ranges. The -v switch will display the original allocation information, since this can be changed by certain allocation functions.

Back to Contents

!pte [Address]

Displays the PTE and/or PDE for a given virtual address or PTE address, along with the PTE status bits.

!pte2va [PTE Address]

Displays the virtual memory address which corresponds to the PTE.

!pfn [Page Frame Address]

Displays information about a given PFN entry.

!memusage [Flags]

Displays summary statistics regarding physical memory.

Flags:

Flag Value	Flag Description
8	General summary information about memory usage.
0	Provides general summary information, along with detailed information about each page. This is the default parameter value.
1/2	Displays summary information about no-write pages, whereas, using the flag value of 2, will provide detailed information about no-write pages.

!sysptes [Flags]

Displays information about system PTEs.

!ptov [PFN Address]

Shows the physical to virtual memory mapping between pages for a given process.

PFN is the first four bits of the Directory Base for the process.

!vtop [0 [Virtual Address]]

Back to Contents

Shows the virtual to physical memory mapping between pages for a given process. This command is essentially the opposite of **!ptov**.

If adding the 0, you'll use the current process context. Otherwise, you'll need to provide the directory base of the process along with it's PFN. Please check the WinDbg documentation for more information.

!memlist [PFN List Address [Flags]]

The debugger will walk the MMPFNLIST structure to verify that there is no corruption to the linked list.

Flags:

Flag Value	Flag Description
0	Walks the zero-paged lists
1	Walks freed page lists

!address [Address [-p]]

If the address has been omitted, then the virtual memory regions for the entire kernel address space are displayed instead. If the -p switch is used, then the physical memory address space will be displayed instead. If an address is provided, then the region which contains that address is displayed only.

!ca [Address | 0 | -1 [Flags]]

Displays information about a particular control region, otherwise if 0 is passed to the command, all of the control regions are displayed.

The flags are used in conjunction with a bitwise OR:

Flags:

Flag Value	Flag Description
1	Displays information about the associated segment.
2	Displays information about the associated sub-section.
4	Displays a list of all mapped views (Windows 7+)
8	Truncates the output into a compact one-line format

10	Displays the file-backed control regions.
20	Displays control regions which are backed by a page file
40	Displays image control regions.

!wsle [Flags [Address]]

Displays information about a working set list entry (WSLE).

Flags:

Flag Value	Flag Description
0	Includes information such as the address of the WSLE, its age, lock status and reference count.
1	Includes information such as the first free WSLE index, the index of the last WSLE and number of valid WSLEs.
2	Includes the number of free WSLEs along with their indexes.

!vad [VAD Root [Flag]]

The VAD root can be obtained by running !process. The command will display the entire VAD (virtual address descriptor) tree by default.

Flags:

Flag Value	Flag Description
0	Displays the entire VAD tree using the specified root.
1	Displays only the VAD root.

Data Structues:

_MI_SYSTEM_VA_TYPE, _HARDWARE_PTE, _MI_SYSTEM_PTE_TYPE, _MMPTE_PROTOTYPE, _MMLISTS, _MMPFNENTRY, _MMPFN, _MMPFNLIST, _MMWSLE

Objects and Handles:

!object [Address | Name | Path [Flags]]

Displays information about an object including it’s object type and header address, along with it’s

reference count. To use the **path** option, you'll need to provide one of the Object Manager's directories, these can be found with the WinObj SysInternals Tool.

Flags:

All the flags are used in conjunction as a Bitwise OR.

Flag Value	Flag Description
0	Displays the object type for the associated object.
1	Displays the object type, name and reference count for the associated object.
8	Displays the object directory which the object resides in. This flag must be used in conjunction with the 0x1 (1) flag.
10	Displays the object header for the associated object.
20	Displays the full path to the object.

!obja [Address]

Displays the object attributes for a given object.

!handle [Handle Address [Flags [Process [Type Name]]]]

Displays information regarding a specific handle or the handles associated to a given process object. Please note that the flags mentioned here are for kernel-mode only. If no handle address is given, then all the handles will be displayed for the current process context. Likewise, if -1 is provided as a handle address, then it will display all of the currently open handles for all processes.

Flags:

Flag Value	Flag Description
0	Displays basic handle information
1	Displays basic object information for the associated handle.
2	Displays free handles, if this flag is not set and no handle address is provided, then the no free handles will be displayed.
4	Display the handle entry from the system-wide handle table, rather than the table associated

	to the process.
5	Causes the debugger to assume that the given handle address is a PID or TID, and therefore will either display the associated process or thread object instead.

!sd [Address]

Displays the security descriptor information for a given object. The address can be found in the Security Descriptor field of the OBJECT_HEADER structure.

Data Structures:

_OBJECT_HEADER, _OBJECT_TYPE, _OBJECT_ATTRIBUTES, _OBJECT_HANDLE_INFORMATION

Processors, Interrupts and IRQLs:

!irql

Displays the IRQL level of a given processor which was saved at the time of the crash.

!idt [IDT Address [-a]]

Displays the IDT for a given processor. The -a switch will display the ISRs for each IDT entry.

You can pass an index of IDT to the command it will display that individual entry.

!ipi [Processor Number]

Displays information about the inter-processor interrupts (IPIs) for the given processor. If no processor index is provided, then all the IPIs will be displayed.

~ [Processor Number]

Changes the debugger context to the specified processor.

!qlocks

Displays information about all global queued spinlocks.

!locks [Flags [Address]]

Displays information about ERESOURCE locks. If no address has been provided, then all the locks will displayed.

Back to Contents

Flags:

Flag Value	Flag Description
-v	Displays detailed information about the lock
-p	Displays performance information
-d	Displays all locks including those which do not have any contention count.

!timer

Displays a list of all the active timer objects.

!dpcs [Processor Number]

Displays the DPC queue for the specified processor.

!pcr [Processor Number]

Displays the processor control region for the specified processor. This command dumps the _KPCR structure. If no processor number has been provided, then the current processor will be used instead.

!pcrb [Processor Number]

Displays the processor control region block for a given processor. If no processor number has been provided, then the current processor will be used instead. This is equivalent to dumping the _KPCRB structure.

!dpcwatchdog

Displays detailed debugging information about DPCs including the DPC queue for each processor.

!frozen

Displays the current state of each processor.

!swd [Processor Number]

Displays the DPC watchdog timeout and DPC times for the processor, unless a process number has been specified.

Back to Contents

!apic

Displays information about a APIC, this includes the APIC mode, local interrupt pins and interrupt lines.

Data Structures:

_KINTERRUPT, _KAPC, _KAPC_STATE, _ERESOURCE, _KSEMAPHORE, _DISPATCHER_HEADER, _FAST_MUTEX, _KEVENT, _KOBJECTS, _KWAIT_BLOCK, _KPCR, _KPRCB, _KDPC, _GROUP_AFFINITY, _KINTERRUPT_MODE, _KINTERRUPT_POLARITY, _IO_INTERRUPT_MESSAGE_INFO, _KIDTENTRY64, _KIDTENTRY

I/O and IRPs:

!irp [Address [Flags]]

Displays information about a I/O packet (IRP) for the provided IRP address.

If an additional flag is provided – can be any number – then additional information will be displayed which includes the owning thread, status of the IRP, the corresponding MDL and the stack locations for the IRP.

!irpfind [-v [Pool Type [Address [Criteria [Data]]]

Searches for a specific IRP matching the criteria.

The -v switch will display additional information about the IRP.

Pool Type:

Flag Value	**Pool Type**
0	Non-paged pool (default value)
1	Paged Pool
2	Special Pool
3	Session Pool

Criteria:

Flag Value	**Flag Description**
arg	Filters the search to IRPs whose stack location contains an argument which is equal to the value provided to the *Data* flag

device	Filters the search to IRPs whose stack location contains a device object which is equal to the value provided to the *Data* flag
fileobject	Filters the search to IRPs whose OriginalFileObject field is equal to *Data*
mdlprocess	Filters the search to IRPs where the Process field is equal to that of *Data*
thread	Finds all the IRPs where the Thread field is equal to that of *Data*
userevent	Finds all the IRPs where the userevent field is equal to that of *Data*

!devobj [Address]

Displays a formatted version of the DEVICE_OBJECT structure.

!drvobj [Address]

Displays a formatted version of the DRIVER_OBJECT structure.

!devnode [Address [Flags]]

Displays information about a device node within the PnP device tree. If no address has been provided, then the root device of the device tree will be displayed instead.

Flags:

Flag Value	**Flag Description**
1	Displays all pending removals of device objects from the device tree.
2	Displays all pending ejects of a device object
0 1	Displays the entire PnP device tree

!devstack [Address]

Displays the device stack for the device object.

!devext [Address [Bus Type Code]]

Displays additional information about the bus a device is attached to.

Back to Contents

Bus Type Codes:

Bus Type Code	Description
ISAPNP	ISA PnP device extension
PCMCIA	PCMCIA device extension
HID	HID device extension

!ioctldecode [IOCTL]

Displays information about a given I/O control code. The IOCTL can be found by using the **!irp** command, and then checking the third argument of the **Args:** field.

Data Structures:

_DEVICE_OBJECT, _DRIVER_OBJECT, _IRP, _MDL, _IO_STATUS_BLOCK, _IO_STACK_LOCATION, _IO_TIMER

System Hardware Information:

!system [machineid | cpuinfo | cpuspeed]

Displays hardware information about a given processor. Please note I've only the three most useful parameters for this extension.

- machineid – displays general BIOS information and motherboard information.
- cpuinfo – displays general information about a given processor
- cpuspeed – displays the current and stock clockspeed for the processor.

!whea

Displays a formatted version of the WHEA_ERROR_RECORD_HEADER structure

!errpkt [Address]

Displays a formatted version of the WHEA_ERROR_PACKET structure

!errrec [Address]

Displays a formatted version of the WHEA_ERROR_RECORD structure

Back to Contents

!tz [Address]

Displays information about a given thermal zone. If no address is provided, then all of the thermal zones will be displayed. This information is gather from the ACPI tables.

Displays hardware information in kelvins.

!tzinfo [Address]

Displays more detailed information about a thermal zone.

!cpuinfo

Displays information about the processor including family, model and stepping.

!cpuid

Displays similar information to the !cpuinfo extension, but will include the clockspeed of the processor as well.

!diskspace [Drive Letter]

Displays the amount of free space for the specified volume. This information may only be available during live debugging sessions.

!pci

Displays the PCI configuration space. Please note that this command is only available during live debugging sessions on a x86-based computer.

!pcitree

Displays all the PCI buses and their attached devices.

Data Structures:

_WHEA_ERROR_RECORD, _WHEA_ERROR_RECORD_HEADER, _WHEA_ERROR_RECORD_HEADER_VALIDBITS, _WHEA_TIMESTAMP, _WHEA_ERROR_SOURCE_TYPE, _WHEA_ERROR_PACKET, _PCI_EXPRESS_UNCORRECTABLE_ERROR_STATUS, _WHEA_ERROR_SEVERITY, _WHEA_ERROR_RECORD_SECTION_DESCRIPTOR, _WHEA_PROCESSOR_GENERIC_ERROR_SECTION, _WHEA_PROCESSOR_GENERIC_ERROR_SECTION_VALIDBITS, _WHEA_XPF_PROCESSOR_ERROR_SECTION, _WHEA_PCIEXPRESS_ERROR_SECTION,

Back to Contents

_WHEA_PCIEXPRESS_DEVICE_TYPE, _WHEA_PCIEXPRESS_DEVICE_ID, _PCI_EXPRESS_AER_CAPABILITY, _WHEA_ERROR_PACKET_V2

Thread Scheduling:

!running [-t] [-i]

Displays a list of all the running threads on all processors.

Flags:

Flag Value	Flag Description
-t	Displays idle processors as well.
-i	Includes the stack trace as well.

!runaway [Flags]

Displays how long a thread has been running for. This can only be used for live debugging sessions and crash dumps created by **.dump /mt**.

Flags:

Flag Value	Flag Description
0	Displays the amount of user-mode time which has been consumed.
1	Displays the amount of kernel-mode time which has been consumed.
2	Displays the amount of time consumed since the creation of the thread.

Registers and Exceptions:

r [Register]

Displays the contents of a processor register. For example, r @eip will display the contents of the eip register.

.trap [Address]

Displays a formatted version of the KTRAP_FRAME structure with the saved registers before the context switch. Please note that the command will change the processor context to the trap frame and not thread.

Back to Contents

.exr [Address]

Displays a formatted version of the EXCEPTION_RECORD structure.

.cxr [Address]

Displays a formatted version of the CONTEXT_RECORD structure, and changes the context to that of the context record itself.

!error [Error Code]

Displays the NTSTATUS code description for the given error code.

!exchain

Lists the exception handlers for the current thread. Please note that this command is only available on x86-based systems.

Examining Memory:

da [Address Range [Lines to Display]]

Displays the memory contents of an array.

dd [Address Range [Lines to Display]]

Displays the memory contents within the given address range.

dc [Address Range [Lines to Display]]

Displays the memory contents as character strings if possible.

du [Address Range [Lines to Display]]

Displays the memory contents as unicode strings if possible.

dps [Address Range [Lines to Display]]

Displays memory contents with symbol information if possible.

dds [Address Range [Lines to Display]]

Similar to **dps** command, however, double word data lengths are used instead.

Back to Contents

Driver Information:

lmvm [Module Name]

Displays detailed information about a driver module. This includes a timestamp.

lm

Displays loaded modules at the time of the crash.

lmstm

Displays detailed information about modules at the time of the crash.

ln [Address]

Displays symbol information for the given address. This is typically used if you wish to know if the address corresponds to a driver or function name.

Power Policy:

!popolicy [Address]

Displays power-related information about the system at the time of the crash. This is a parsed from the SYSTEM_POWER_POLICY data structure.

If the address is omitted, then the address stored in the *nt!PoPolicy* is used instead.

!pocaps

Displays information regarding the power capabilities of the system. This is ideal for checking if a driver is attempting to use an unsupported power state.

!poaction

Displays a list of outstanding power IRPs along with any completed IRPs. The command will also show the current power state of the system along with the individual power states of devices. It will also show the order in which devices are powered down during a shutdown.

!podev

Displays power related information about a given PnP device object.

Back to Contents

!poreqlist

Displays a list of outstanding power IRPs created using the *PoRequestPowerIrp* function. This function will create a power IRP and associate it to the top of device stack for a given device object.

!powertriage

Displays detailed power information about various devices. This command is a combination of several other commands.

!pnptriage [Address]

Parses and displays the _TRIAGE_9F_PNP structure.

Data Structures:

_SYSTEM_POWER_STATE, _DEVICE_POWER_STATE, _SYSTEM_POWER_POLICY, _TRIAGE_9F_POWER, _TRIAGE_9F_PNP, _POP_FX_COMPONENT, _POP_FX_DEVICE, PDC_CLIENT_TYPE, PDC_NOTIFICATION_TYPE, _PDC_NOTIFICATION_CLIENT, PDC_NOTIFICATION_CONTROL, _PDC_RESILIENCY_CLIENT, PDC_RESILIENCY_TYPE, _PDC_ACTIVATOR_CLIENT, PDC_ACTIVITY_TYPE, _PDC_ACTIVATION_STATS, _PDC_14F_TRIAGE

Local Inter-Process Calls (ALPCs):

!alpc /lpp [Process Address]

Displays all connections for that process. This includes ports created by the process and ports which the process is currently connected to.

!alpc /p [Port Address]

Displays information regarding the specified port. This includes the port, server communication port, client communication port, connection port and message queue information.

!alpc /m [Message Address]

Displays information in relation to the specified message.

Data Structures:

_CLIENT_ID, _LPCP_PORT_OBJECT, _LPCP_PORT_QUEUE, _LPCP_NONPAGED_PORT_QUEUE, _KALPC_MESSAGE, _LPCP_MESSAGE, _PORT_MESSAGE

Back to Contents

Windows Registry:

!reg hivelist

Displays all the hives in the registry along with information about each hive.

!reg dumppool

Displays paged pool consumption for each registry hive.

!reg viewlist [Hive Address]

Displays memory mapped view information for a selected hive.

!reg freebins [Hive Address]

Displays the free bins for a specified registry hive.

!reg openkeys [Hive Address or 0]

Displays all the open keys for a specified hive, unless **0** is specified instead, which will display all the open keys in the registry.

!reg findkcb [Registry Key Path]

Displays the KCB address for the given registry key path.

!reg kcb [KCB Address]

Displays information about the key control block for the given registry key.

!reg cellindex [Hive Address [Cell Index]]

Displays a given cell index for the registry hive.

!reg querykey [Key Path]

Displays the subkeys and values for the specified key.

Data Structures:

Back to Contents

_CMHIVE, _HHIVE, _CM_CELL_DATA, _CM_KEY_CONTROL_BLOCK, _CM_KEY_HASH, _HMAP_DIRECTORY, _HMAP_TABLE, _HMAP_TABLE_ENTRY

Process Heaps:

!heap

Displays the process heap information including any leaks. This command extension also supports NT as well as segment heaps. There is large number of different switches/flags which you can use, so I recommend that you also check the WinDbg documentation.

Flags:

Flag Value	Flag Description
-stat (NT Heap only)	Provides a summary of the active heaps.
-m (NT Heap only)	Shows the segment entries for a given heap. The address of the heap to be examined must be provided.
-s	Displays summary information for all heaps within a given process.
-h [Heap Address]	Displays information about a particular heap.
-l	Checks for any leaked blocks in the heap.

Data Structures:

_HEAP, _HEAP_LOCK, _HEAP_TUNING_PARAMETERS, _DPH_BLOCK_INFORMATION

Linked Lists:

dl [b [Address [MaxCount [Size]]]

Parses a doubly or single linked list. The **b** parameter will transverse and display the doubly linked list in reverse.

The address is the starting address of the linked list. The number of nodes to be displayed can be controlled by specifying a whole number for the max count parameter.

!validatelist [Address]

Ensures that a doubly linked list is valid by ensuring that the flinks and blinks correspond with each other.

Back to Contents

!aflink [Address [Count]]

Transverses and displays a doubly linked list using the flinks. The address is the starting address to transverse with. The count determines the number of nodes to display with the maximum number being 32 entries.

!dblink [Address [Count]]

Transverses and displays a doubly linked list using the blinks. The address is the starting address to transverse with. The count determines the number of nodes to display with the maximum number being 32 entries.

Data Structures:

_LINKED_LIST, _LIST_ENTRY

Managing WinDbg Extensions:

!load [Name]

Loads the custom debugger extension by its name.

!unload [Name]

Unloads the custom debugger extension by its name.

.chain

Displays a list of the currently loaded debugger extensions for WinDbg.

.extpath [Path [+]]

Displays the current extension path for which WinDbg will search for command extensions. If a directory path is provided, then the extension path will be set to that path for the duration of the debugging session.

Otherwise, if the + switch is used along with the path, then it will be appended to the current extension path for the duration of the session. You can use a semi-colon (;) to chain several paths together in one statement.

Debugging WinDbg Symbols:

.reload

Back to Contents

Reloads the symbols from the symbol store, this is useful if there are symbol errors or you have added additional symbol files (.pdb)

!sym noisy

Displays debugging information regarding the loading of symbols.

!sym quiet

Stops the debugger from displaying debugging information during symbol loading.

!chkimg [Module Name] Flags]]

Verifies the symbol of a binary image file within the dump file against the symbol stored within the Microsoft Symbols store or local symbol store. This is used to detect if a binary file is corrupt.

Flags:

Flag Value	Flag Description
-f	This will fix any corruption found within the image, by transferring the symbols from the symbol store to the dump file
-v	Displays verbose information.
-d	Shows the number of mismatch errors.

.symfix

Sets the symbol path for the debugger to Microsoft's symbol server.

.sympath [Symbol Path [+]]

Displays the current symbol path for which WinDbg will search for symbols. If a symbol path is provided, then the symbol path will be set to that path for the duration of the debugging session.

Otherwise, if the + switch is used along with the path, then it will be appended to the current symbol path for the duration of the session. You can use a semi-colon (;) to chain several paths together in one statement.

!lmi [Module Address/Module Name]

Displays detailed information about a given module. You can specify the base address of the module or provide the module name. The command will display the status of the symbols

associated to the module, along with the module's image name.

Miscellaneous:

.bugcheck

Displays the bugcheck code and its parameters.

.dumpdebug

Parses the DUMP_HEADER64 and TRIAGE_DUMP64 data structures.

vertarget

Displays operating system information from the dump file.

.time

Displays the time which the crash dump was generated.

x [Options [Module!Symbol]]

This command will search for symbols which match the provided search pattern. There is a large number of switches you can provide, however, please consult the WinDbg documentation for more details. The search pattern uses the Wildcard syntax. The following table will explain some of the options:

Wildcard Syntax:

Syntax	Description	Example
*	Represents 0 or more characters.	**x** nt!*time* **x** nt!*time
?	Represents a single character.	**x** nt!*time?
[-]/[]	Represents any single character within the list. The hyphen can be added to match any single character within the range.	**x** nt!*[time]? **x** nt!*[ti-me]
#	Matches 0 or more preceding characters.	**x** nt!*ca#n
+	Matches 1 or more preceding characters.	**x** nt!Vf+*

Back to Contents

You can find all the symbols for a particular module by using **moduleName!***.

!dh [Module Name/Address [Flags]]

Displays the directories and sections of a PE file.

Flags:

Flag Value	Flag Description
-f	Displays the file headers only
-s	Displays the section headers only.
-a	Displays all the header information.

.fnent [Address]

Displays the function table entry for the given function address. This includes the unwind metadata used for building the call stack.

Driver Verifier:

!deadlock

Displays information about a deadlock detected by Driver Verifier. Using **!deadlock 1** will display the stacks of the deadlocked threads.

You must enable the Deadlock Detection option in Driver Verifier for this extension to work.

!verifier [Flags]

Displays summary status information regarding Driver Verifier if all flags are omitted. There is a quite a considerable amount of different flags for Driver Verifier, and therefore I would recommend that you consult the documentation for more details.

Flags:

Flag Value	Flag Description
0x1	Displays the names of all the drivers being verified.
0x200	Displays the Critical Region log.
0x8	Displays the IRQL changes made by drivers being tracked by Driver Verifier.

Back to Contents

!ruleinfo [Rule Id [Rule State [Substate]]]

Displays the description of the violated DDI compliance rule. The rule id, state and sub-state will be provided in the Stop 0xC4 bugcheck parameters.

Disassembly Commands:

u [Address/Function Name [Range]]

Disassembles 9 instructions from the start address, unless an address range has been provided.

ub [Address/Function Name [Range]]

Disassembles 9 instructions before the start address, unless an address range has been provided.

Atom Tables:

!atom [Address]

Displays the atom table with the given address, otherwise displays the atom table for the current process.

!gatom

Displays the global atom table.

For additional information please consult the WinDbg documentation.