The Windows Concept Journey

Version 6.0
March-2025

By Dr. Shlomi Boutnaru

Created using Craiyon, AI Image Generator

Table of Contents

Introduction

When starting to learn Windows I believe that they are basic concepts that everyone needs to know about. Because of that I have decided to write a series of short writeups aimed at providing a basic explanation for fundamental concepts which are part of the Windows operating system.

Overall, I wanted to create something that will improve the overall knowledge of Windows in writeups that can be read in 1-3 mins. I hope you are going to enjoy the ride.

Lastly, you can follow me on twitter - @boutnaru (https://twitter.com/boutnaru). Also, you can read my other writeups on medium - https://medium.com/@boutnaru. Lastly, You can find my free eBooks at https://TheLearningJourneyEbooks.com.

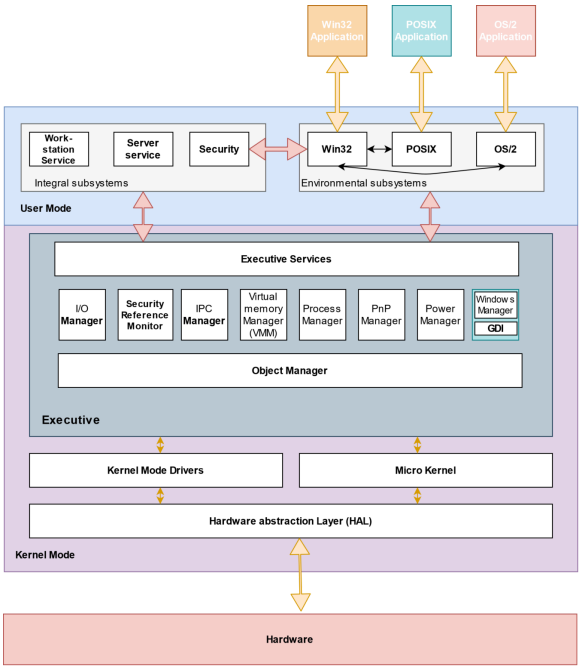
Lets GO!!!!!!

Windows NT Architecture

As with other operating systems also the “Windows NT Architecture” is based on layer design. The layer design has two main components: “User Mode” and “Kernel Mode”. The operating system is reentrant multitasking and preemptive, which supports both uniprocessors and SMP (Symmetric Multiprocessors). Moreover, since Windows XP there is both a 32-bit version and 64-bit version of the OS[1].

Overall, the “User Mode” of a Windows NT system is composed of executables (which are then executed as system processes) and DLLs (Dynamic Link Libraries) - as shown in the diagram below. Also, Windows NT has three subsystems included: Win32 subsystem, POSIX subsystem (which was replaced by WSL) and the OS/2 subsystem (which is deprecated since Windows XP)[2].

Lastly, the “Kernel Mode” is also made up of different components (as shown in the diagram below): “Executive” (Object Manager/IO Manager/SRM/Memory Manager/etc), “Kernel”, “Drivers” and the “HAL” (Hardware Abstraction Layer) [3].



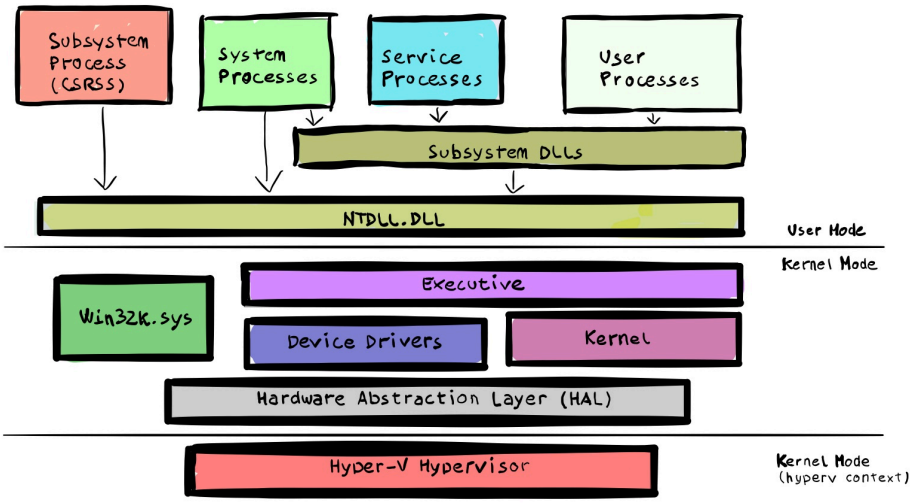
[1] https://en.wikipedia.org/wiki/Architecture_of_Windows_NT
[2] https://blog.certcube.com/the-nt-architecture-of-windowss/
[3] https://www.cs.fsu.edu/~zwang/files/cop4610/Fall2016/windows.pdf

Windows NT Architecture with the Hypervisor Layer

Since Windows 10 there is an addition to the general "Windows NT Architecture"[4]. The new architecture includes an additional hypervisor layer (which does not to be enabled) - as shown in the screenshot below[5].

Overall, this new layer is below the HAL (Hardware Abstraction Layer). This layer is completely separated from the whole virtual machine. Thus, Windows uses the Hyper-V hypervisor as an additional privilege level to hide away security functionality[6].

Lastly, the hypervisor as part of the Windows architecture is a fundamental component of the "VBS" (Virtual Based Security) feature. VBS provides basic functionality needed by different security capabilities in Windows such as: HVCI (Hypervisor Enforced Code Integrity) and "Credentials Guard"[7].



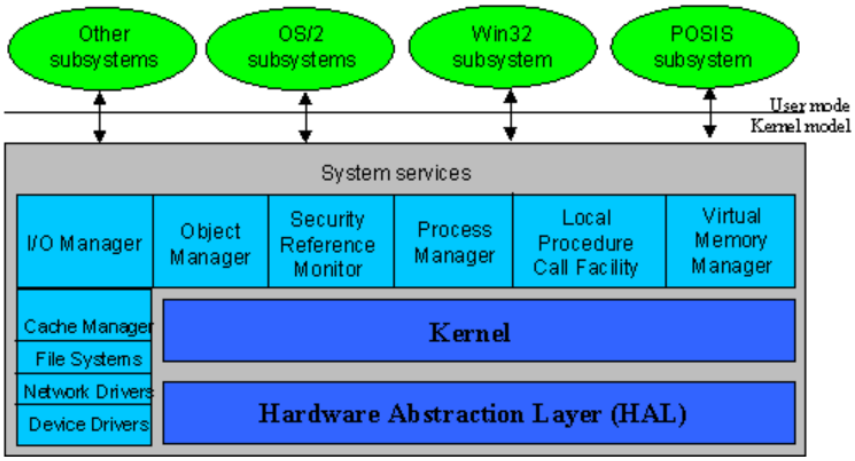
[4] https://medium.com/@boutnaru/the-windows-concept-journey-windows-nt-architecture-e2665908578c
[5] https://www.matteomalvica.com/minutes/windows_kernel/
[6] https://stl-tec.de/tutorials/WinReverseEng/windowsArchitecture/3
[7] https://learn.microsoft.com/en-us/windows/security/hardware-security/enable-virtualization-based-protection-of-code-integrity?tabs=security

Executive (The Windows Executive)

As detailed in the "Windows NT Architecture"[8] the kernel mode of the Windows operating system is divided into different parts, one of them is the "Executive". The "Executive" is implemented as part of the "ntoskrnl.exe"[9], which is located at "%windir%\system32".

Overall, among the different tasks provided by the "Executive" we can find I/O management, object management, security management and processes management. Those different tasks are scattered across several subsystems like: "Object Manager", "Memory Manager", "Cache Manager", "Configuration Manager", "I/O Manager", "PnP Manager", "Power Manager", "SRM" (Security Reference Monitor), "Process Manager" and more[10].

Lastly, callable routines exposed by the "Executive" are also called "services" (aka "system services"). Thus, we can say the "Executive" is a family of software components that provide basic operating system services to the protected subsystems and to each other. The components are independent and communicate through controlled interfaces - as shown in the diagram below[11] .By the way, we can check out the reference implementation of the "Executive" services as part of reactOS[12].



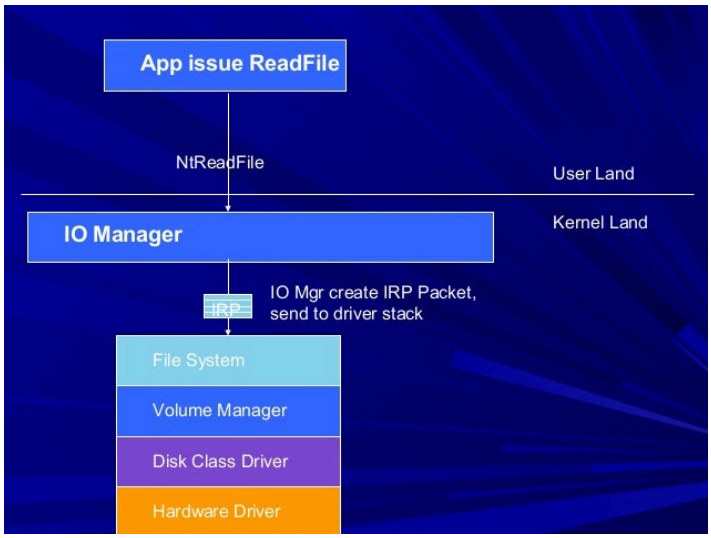
[8] https://medium.com/@boutnaru/the-windows-concept-journey-windows-nt-architecture-e2665908578c
[9] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[10] https://learn.microsoft.com/en-us/previous-versions//cc768129(v=technet.10)
[11] https://wwwdisc.chimica.unipd.it/luigino.feltre/pubblica/unix/winnt_doc/rk/rk_ntsvrunix_sjuq.html
[12] https://github.com/reactos/reactos/tree/master/ntoskrnl

I/O Manager (Input/Output Manager)

The “I/O Manager” is used for managing hardware devices and providing interfaces for applications to access/use those hardware devices. Thus, the “I/O Manager” acts as a connector between user-mode applications and the relevant hardware devices. Also, the “I/O Manager” works in conjunction with the “PnP Manager” and the “Power Manager”[13].

Overall, the “I/O Manager” is a Windows subsystem that performs the following tasks: enforcing security/access control as part of I/O operations, managing sync/async I/O tasks, creating/managing IRPs (I/O request packets) and routing IRPs to the relevant device drivers[14] - a general flow is shown in the diagram below[15]. It is important to understand that I/O operations are layered (driver stack) - shown in the screenshot below. Moreover, the “I/O Manager” describes a standard of functions for developers[16]

Lastyl, the functions that are provided by kernel-mode component of the “PnP Manager” are implemented as part of binary “ntoskrnl.exe”[17] are prefixed with “Io/Iop/Iov”[18]. For a reference implementation of them we can checkout the source code of ReactOS[19].



[13] https://www.microsoftpressstore.com/articles/article.aspx?p=2201309
[14] https://networkencyclopedia.com/i-o-manager/
[15] https://www.slideshare.net/SisimonSoman/windows-io-manager
[16] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/overview-of-the-windows-i-o-model
[17] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[18] https://codemachine.com/articles/ntoskrnl_component_list.html
[19] https://github.com/reactos/reactos/tree/master/ntoskrnl/io/iomgr

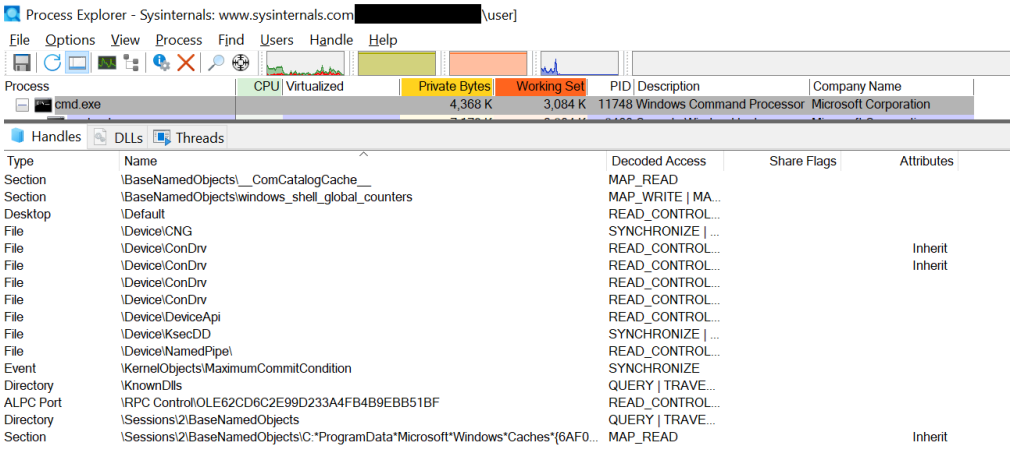
Object Manager

As part of the Windows operating system there is a single “Object Manager” that maintains all the objects[20]. Among the tasks the “Object Manager” does are: creating objects, verifying that a process has a right to use an object, creating object handles[21] and returning them to the caller, maintaining resource quotas, duplicating handles and closing handles[22].

Overall, Windows has more than 25 types of objects. Some examples are: files, devices, threads, processes, events, mutexes, jobs, registry keys, sections, access tokens - as shown in the screenshot below taken from “Process Explorer”. The kernel routines which provide a direct interface with the “Object Manager” are prefixed with “Ob”[23]. Some examples are: “ObGetObjectSecurity”[24] and “ObReferenceObjectByHandle”[25].

Moreover, we can go over the reference implementation of the “Object Manager” as part of ReactOS[26]. There is also the internal header file[27].

Lastly, we can summarize the “Object Manager” as being responsible for keeping track of all the resources in Windows. It also provides a way for applications to access and manage those resources in a secure and efficient way[28].



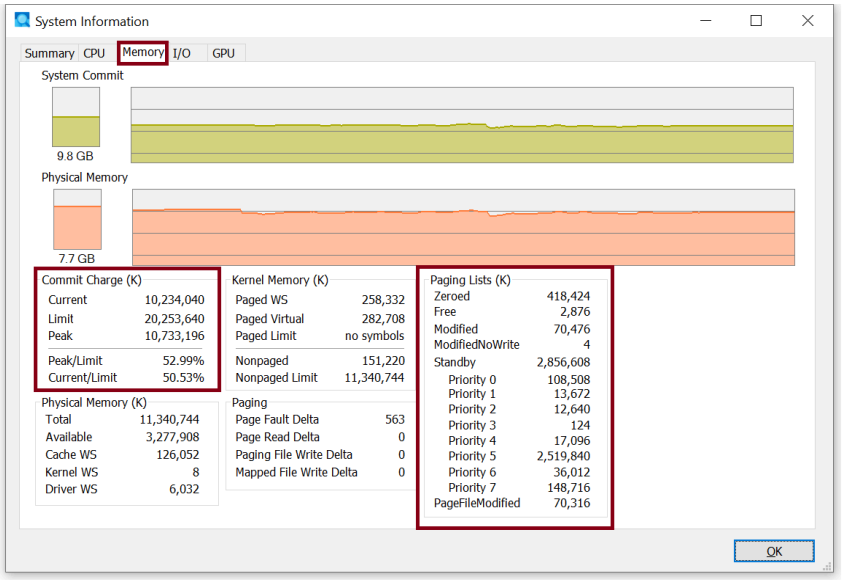
[20] https://medium.com/@boutnaru/windows-objects-2c289da600bf
[21] https://medium.com/@boutnaru/windows-handles-594b36c39d2f
[22] https://learn.microsoft.com/en-us/windows/win32/sysinfo/object-manager
[23] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-object-manager
[24] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-obgetobjectsecurity
[25] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-obreferenceobjectbyhandle
[26] https://github.com/reactos/reactos/tree/master/ntoskrnl/ob
[27] https://github.com/reactos/reactos/blob/master/ntoskrnl/include/internal/ob.h
[28] https://www.i.u-tokyo.ac.jp/edu/training/ss/lecture/new-documents/Lectures/01-ObjectManager/ObjectManager.pdf

Memory Manager

The goal of the "Memory Manager" is to handle/manage the physical memory of the current system, which is RAM based (Random Access Memory). Among the tasks performed by the "Memory Manager" we can find: allocation\deallocation of virtual memory[29] and supporting COW (copy on write), shared memory and memory mapped files[30].

Overall, the basic atoms of the "Memory Manager" are pages. They are handled in different paging lists: "Zeroed", "Free", "Modified", "Modified No Write" and "Standby" (in different priorities) - as shown in the screenshot below, taken from "Process Explorer"[31].

Lastly, memory pages have different characteristics such as: "Committed", "Reserved", "Paged Pool", "Nonpaged Pool" - more information about those in future writeups. Also, for a reference implementation of the "Memory Manager" we can checkout ReactOS[32].



[29] https://medium.com/@boutnaru/linux-memory-management-part-1-introduction-896f376d3713
[30] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-memory-manager
[31] https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer
[32] https://github.com/reactos/reactos/tree/master/ntoskrnl/mm

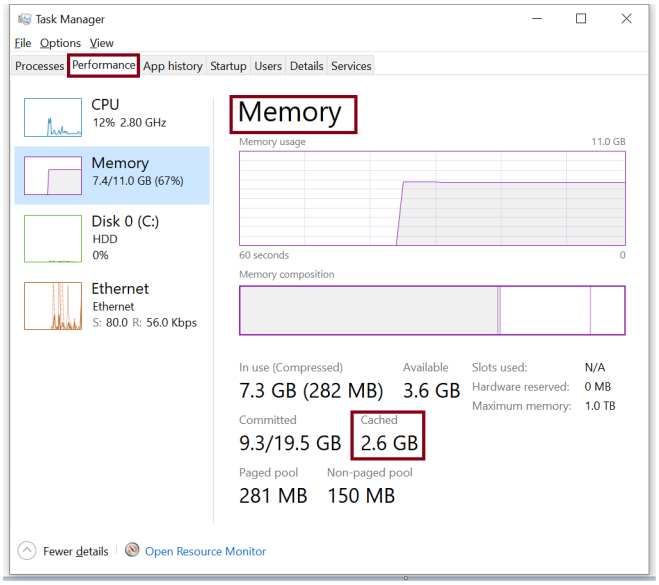
Cache Manager

The “Cache Manager” is Windows works together with the “Memory Manager”[33] in order to provide data caching for local/network based file systems[34]. Caching is an important performance optimization that every modern operating system performs. For example file systems uses the “Cache Manager” for caching metadata like the “File Allocation Table” in FAT or the “Master File Table” (MFT) in NTFS[35].

Overall, “Cache Manager” provides the following capabilities: access methods for pages representing data of opened files, automatic “lazy write” (aka asynchronous write behind), automatic asynchronous read ahead and IRP bypass (fast I/O)[36].

Thus, we can think about the “Cache Manager” as a virtual memory region in the kernel address space that maps file data to provide quick access to them in the future, which is accessed by the file system driver and/or the virtual memory manager[37]. We can see the amount of memory used by the cache using “taskmgr.exe”[38] - as shown in the screenshot below.

Lastly, the “Cache Manager” functions implemented as part of “ntoskrnl.exe”[39] and prefixed with “Cc”. By the way, for a reference implementation of the “Cache Manager” we can checkout ReactOS[40].



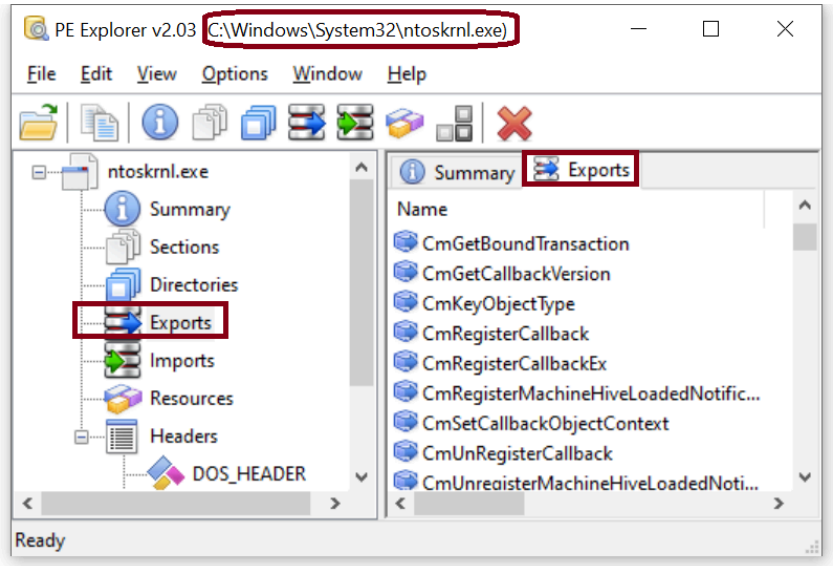
[33] https://medium.com/@boutnaru/the-windows-concept-journey-memory-manager-777660e9b85d
[34] https://www.oreilly.com/library/view/windows-internals-fifth/9780735625303/ch10.html
[35] https://www.itprotoday.com/it-infrastructure/inside-the-cache-manager
[36] https://www.i.u-tokyo.ac.jp/edu/training/ss/lecture/new-documents/Lectures/15-CacheManager/CacheManager.pdf
[37] https://www.linkedin.com/pulse/inside-windows-cache-manager-artem-baranov
[38] https://medium.com/@boutnaru/the-windows-process-journey-taskmgr-exe-task-manager-005753dbcf3a
[39] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[40] https://github.com/reactos/reactos/tree/master/ntoskrnl/cc

CM (Configuration Manager)

The goal of the "Configuration Manager" is to manage the registry[41]. Think about a case in which when writing a kernel mode driver we need to know about changes made to the registry. In that case we can register a callback (CmRegisterCallback\CmRegisterCallback function as an exemple) that is triggered when there is a data change in the relevant registry location[42].

Overall, we can say that the "Configuration Manager" is the Windows subsystem which manages the registry. One example is the need of the "Configuration Manager" of knowing if a key is already open. This is for providing the same handle to other requesters. The reason is to ensure all application are referencing the same data[43].

Lastly, the functions provided by "Configuration Manager" are implemented by "ntoskrnl.exe"[44] and prefixed with "Cm/Cmp"[45] - as shown in the screenshot below, which was taken using "PE Explorer"[46]. For a reference implementation of them we can checkout the source code of ReactOS[47].



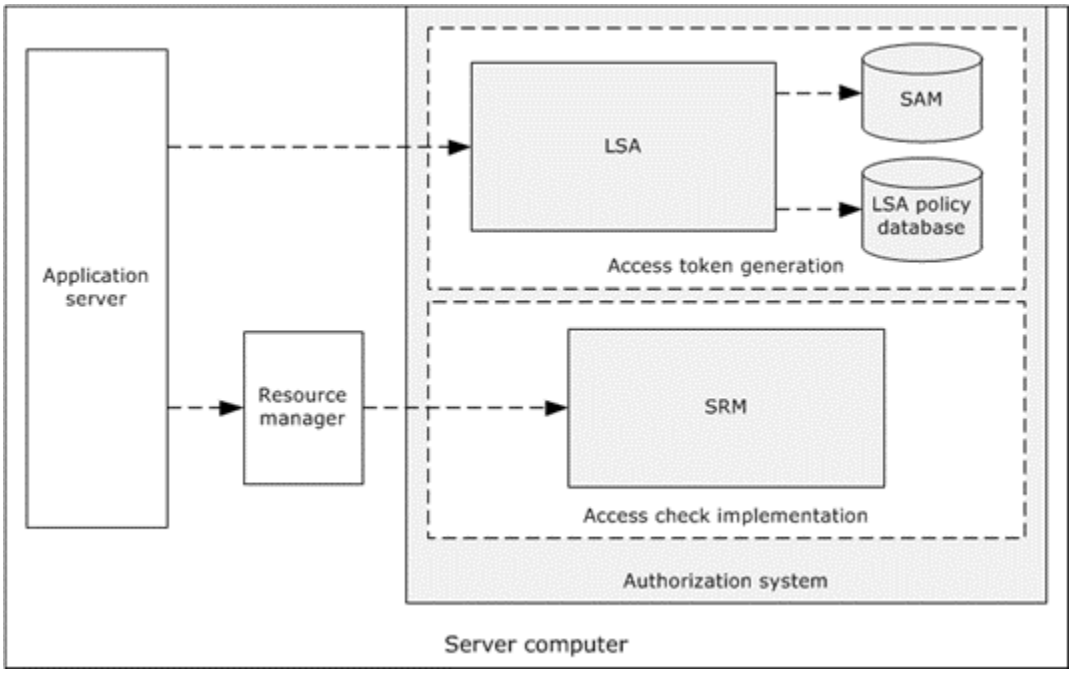
[41] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[42] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-configuration-manager
[43] https://www.sciencedirect.com/science/article/pii/S1742287608000297#bib3
[44] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[45] https://codemachine.com/articles/ntoskrnl_component_list.html
[46] https://github.com/zodiacon/PEExplorerV2
[47] https://github.com/reactos/reactos/tree/master/ntoskrnl/config

SRM (Security Reference Monitor)

SRM (Security Reference Monitor) is a component that is part of the Windows executive (stored in %systemroot%\System32\ntoskrnl.exe). SRM is responsible for implementing the authorization system (together with LSA as shown in the diagram below). Also, SRM implements the access check algorithm[48].. This means it checks the access to different resources by getting the access token[49] of the subject and comparing it to the ACEs (Access Control Lists) in the security descriptor of the securable object[50].

Moreover, the routines that provide a direct interface with the SRM are those prefixed with "Se"[51]. An example of such function is: "SeAccessCheck" which determines if the requested access to an object can be granted[52]. If we want we can go over a reference implementation of "SeAccessCheck" as part of ReacOS[53].

Lastly, we can say that the "Object Manager" uses SRM to check if a specific process/thread has the proper rights to execute a certain action on an object. Also, it is part of the flow when implementing auditing functionality when objects are being accessed[54].



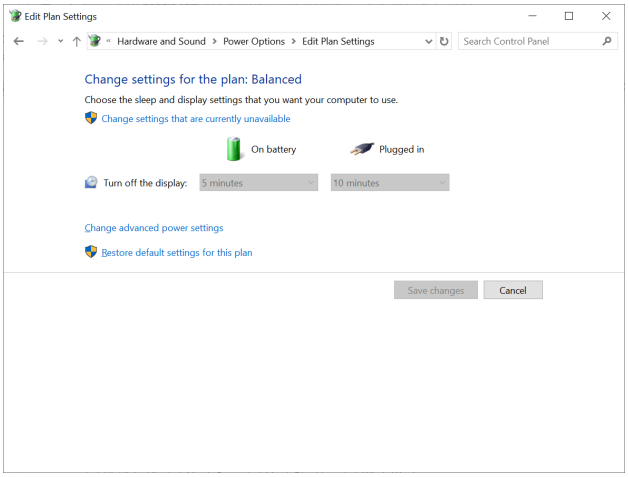
[48] https://learn.microsoft.com/en-us/openspecs/windows_protocols/ms-azod/d28d536d-3973-4c8d-b2c9-989e3a8ba3c5
[49] https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64
[50] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[51] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-security-reference-monitor
[52] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-seaccesscheck
[53] https://github.com/reactos/reactos/blob/master/ntoskrnl/se/accesschk.c#L1966
[54] https://cs.gmu.edu/~menasce/osbook/nt/sld034.html

Power Manager

The “Power Manager” is one of the “Executive”[55] components. It is used for managing the power state of support hardware like external monitors[56]. Thus, it deals with power events such as: stand-by, power-off and hibernate, while notificing relevant drivers about those changes using power IRPs[57].

Overall, the architecture of the Windows power management provides different capabilities to the user such as: providing quiet operation (like powering down devices which are not used to reduce noise), minimal startup/shutdown delays (like leveraging a sleep state) and extending battery life and/or energy saving[58]. We can configure those settings using the “Power Options” as part of the “Control Panel”[59] - as shown in the screenshot below.

Lastly, the functions that are provided by kernel-mode component of the “Power Manager” are implemented as part of binary “ntoskrnl.exe”[60] and prefixed with “Po/Pop”[61]. For a reference implementation of them we can checkout the source code of ReactOS[62].



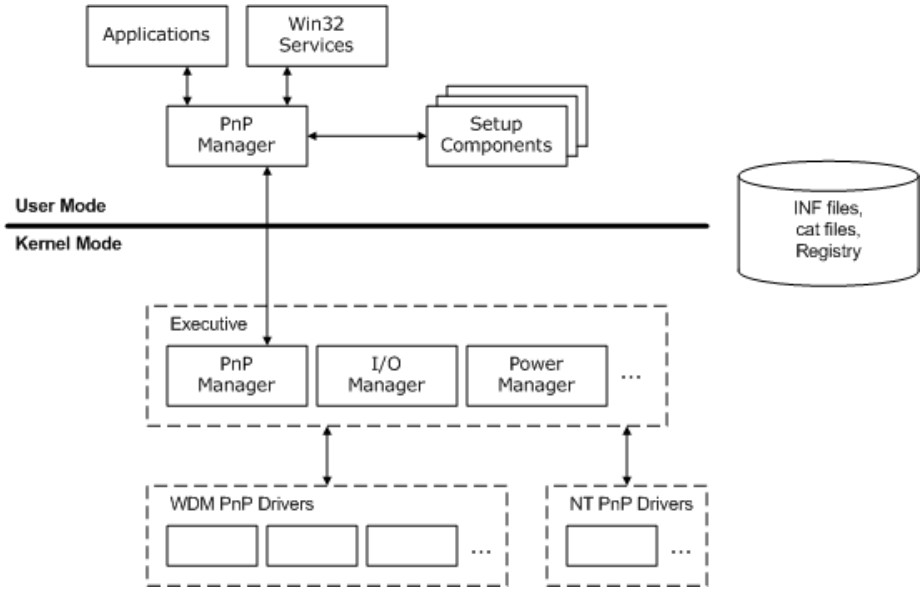
[55] https://medium.com/@boutnaru/the-windows-concept-journey-executive-the-windows-executive-25bb70a40789
[56] http://shamrock-security.com/exploring-operating-systems-the-windows-executive
[57] https://en.wikipedia.org/wiki/Architecture_of_Windows_NT
[58] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/introduction-to-power-management
[59] https://medium.com/@boutnaru/the-windows-concept-journey-control-panel-34bf84ca7ff0
[60] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[61] https://codemachine.com/articles/ntoskrnl_component_list.html
[62] https://github.com/reactos/reactos/tree/master/ntoskrnl/po

PnP Manager (Plug and Play Manager)

The “PnP Manager” (Plug and Play Manager) is used to provide the Windows operating system the following capabilities: installing new devices (with relevant driver package), processing addition/removal of devices which the system is running and device enumeration/detection while the operating system is booting. Also, the “PnP Manager” holds the “DeviceTree” that tracks the devices present currently on the system[63].

Overall, it is important to understand that the “PnP Manager” has both a kernel mode component (as part of the Executive) and a user mode component - as shown in the diagram below. Windows support as part of the plug and play framework physical/logical/virtual devices[64].

Lastly, the functions that are provided by kernel-mode component of the “PnP Manager” are implemented as part of binary “ntoskrnl.exe”[65] and prefixed with “Pp/Pnp/Pi/Pip”[66]. For a reference implementation of them we can checkout the source code of ReactOS[67].



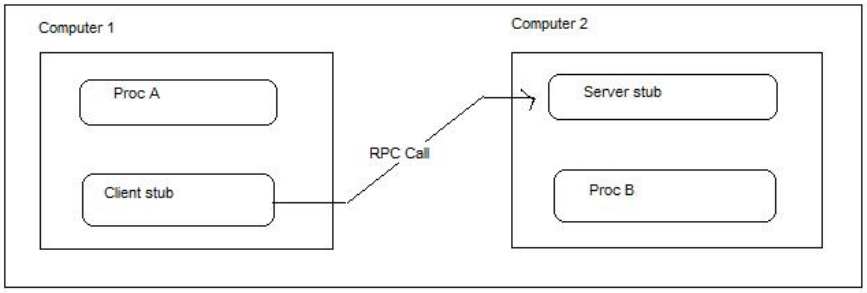
[63] https://learn.microsoft.com/en-us/windows-hardware/drivers/install/pnp-manager
[64] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/pnp-components
[65] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[66] https://codemachine.com/articles/ntoskrnl_component_list.html
[67] https://github.com/reactos/reactos/tree/master/ntoskrnl/io/pnpmgr

IPC Manager (Inter-Process Communication Manager)

IPC stands for “Inter-Process Communication”. Thus, the “IPC Manager” is used for handling communication between clients and servers which are part of the Executive[68]. We can divide the type of communication into two clusters: LPC (Local Procedure Call) and RPC (Remote Procedure Call) - as shown in the diagram below[69]. The first is relevant for clients and servers running on the same system. The second, supports the case in which the clients and servers are not running on the same system[70].

Overall, Windows supports different IPC mechanisms like (but not limited to): anonymous pipes, named pipes, LPC, ALPC, RPC, Windows messages, DDE (which is based on Windows messages), sockets and mailslots[71].

Lastly, the functions that are provided by kernel-mode component of the “IPC Manager” are implemented as part of binary “ntoskrnl.exe”[72]. For a reference implementation of them we can checkout the source code of ReactOS[73].



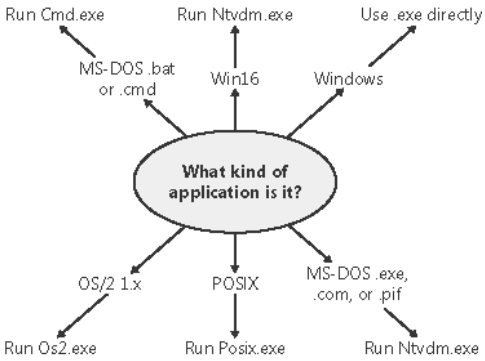
[68] https://medium.com/@boutnaru/the-windows-concept-journey-executive-the-windows-executive-25bb70a40789
[69] https://796t.com/content/1541975242.html
[70] https://blog.certcube.com/the-nt-architecture-of-windowss/
[71] https://csandker.io/2021/01/10/Offensive-Windows-IPC-1-NamedPipes.html
[72] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[73] https://github.com/reactos/reactos/tree/master/ntoskrnl/lpc

Process Manager

The “Process Manager” is one of the “Executive”[74] components. It is used for managing the creation/termination of processes/threads[75]. Also, the “Process Manager” is used for controlling resource allocation for processes and keeping track of information about both processes and threads[76].

Moreover, as opposed to Linux/Unix in which need to call two different system calls in order to spawn a new process: “fork”[77] and “execve”[78] under Windows we just need “CreateProcess” API[79].

Lastly, the functions that are provided by kernel-mode component of the “Process Manager” are implemented as part of binary “ntoskrnl.exe”[80] are prefixed with “Ps/Psp”[81]. For a reference implementation of them we can checkout the source code of ReactOS[82]. By the way, based on the file type and its properties when calling “CreateProcess” Windows finds the appropriate Windows image that will run the executable file[83] - as shown in the diagram below.



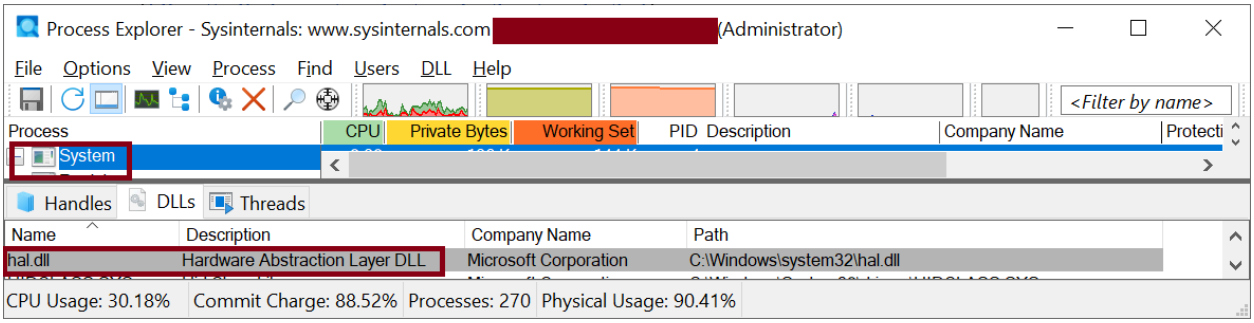
[74] https://medium.com/@boutnaru/the-windows-concept-journey-executive-the-windows-executive-25bb70a40789
[75] https://dlab.epfl.ch/wikispeedia/wpcd/wp/w/Windows_2000.htm
[76] https://efreidoc.fr/L3/Operating%20System/Cours/PDF/2010-11/2010-11_cours_11_windows_object-manager-and-process-management_op.pdf
[77] https://man7.org/linux/man-pages/man2/fork.2.html
[78] https://man7.org/linux/man-pages/man2/execve.2.html
[79] https://www.coresecurity.com/core-labs/articles/creating-processes-using-system-calls
[80] https://medium.com/@boutnaru/the-windows-process-journey-ntoskrnl-exe-nt-kernel-system-7abf8c5a4ed4
[81] https://codemachine.com/articles/ntoskrnl_component_list.html
[82] https://github.com/reactos/reactos/tree/master/ntoskrnl/ps
[83] https://flylib.com/books/en/4.491.1.52/1/

HAL (Hardware Abstraction Layer)

The “HAL” is one of the basic blocks of the “Windows NT Architecture”[84]. HAL stands for “Hardware Abstraction Layer”, it is used to abstract the low-level hardware details from drivers and/or the operating system. This is due to the fact Windows needs to support different hardware configurations[85]. The functions which interface to the HAL are prefixed with “Hal” such as: “HalAllocateCommonBuffer” and “HalSetBusData”[86].

Overall, the HAL provides an interface between the hardware and the OS. This allows the kernel to stay the same while different hardware devices are in use. Examples of use cases in which the HAL is leveraged are: interrupt handling/routing, I/O ports access, accessing physical memory and more[87].

Lastly, the HAL is implemented in files like “hal.dll” which is loaded/mapped to the kernel[88]. We can see that by viewing the “DLLs” tab of the “System” process (PID 4) using “Process Explorer”[89] - as shown in the screenshot below. For a reference implementation of HAL we can check out the source code of ReactOS[90].



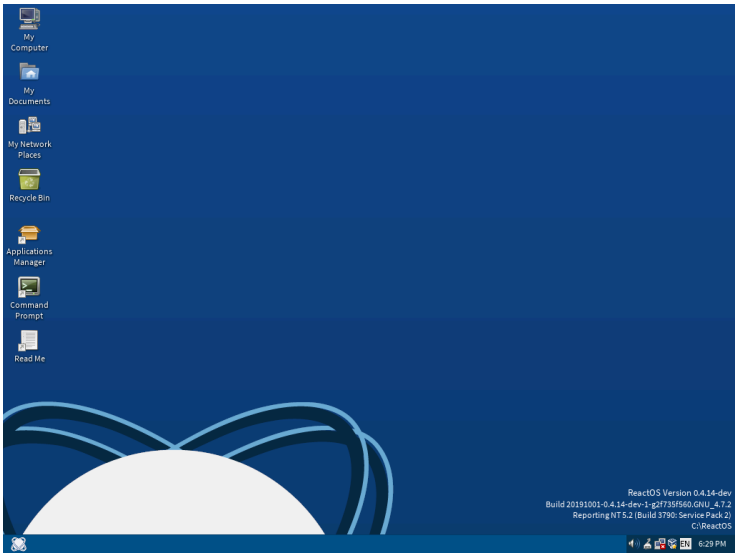
[84] https://medium.com/@boutnaru/the-windows-concept-journey-windows-nt-architecture-e2665908578c
[85] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/windows-kernel-mode-hal-library
[86] https://learn.microsoft.com/en-us/previous-versions/windows/hardware/kernel/ff546644(v=vs.85)
[87] https://library.mosse-institute.com/articles/2022/12/windows-internals-hal.html
[88] https://medium.com/@boutnaru/the-windows-concept-journey-not-all-dlls-are-loaded-mapped-in-user-mode-b6cd8cb48db8
[89] https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer
[90] https://github.com/reactos/reactos/tree/master/hal

ReactOS

In general ReactOS is a free/open source implementation of Windows - as shown in the screenshot below. The goal of the project is to allow running Windows applications/drivers in a trusted open source environment. Also, the UI has a similar look and feel as the Windows operating system[91] - as shown in the screenshot below.

Overall, we can summarize the architecture of ReactOS to the following layers: “ReactOS Applications” (which are written in C\C++ leveraging the Win32 API), “System Processes” (think about “smss.exe, “winlogon.exe”, “lsass.exe” and more), “Shell and Explorer”, “Win32 API”, “Drivers” and the “NT kernel”[92].

Lastly, we can use ReactOS source code[93] as a reference implementation of different Windows components that we can research. Also, the main goal of the ReactOS project is to provide an operating system which is binary compatible with Windows. Thus, Windows applications and drivers run as they would on a Windows system[94]



[91] https://reactos.org/
[92] https://reactos.org/architecture/
[93] https://github.com/reactos/reactos
[94] https://distrowatch.com/table.php?distribution=reactos

WRK (Windows Research Kernel)

WRK (Windows Research Kernel) is a portion of the source code of "Windows XP"\"Windows 2003 Server" service pack 1 (2005 edition) - as shown in the screenshot below. The main usage of WRK was in universities\academies\scientific centers for investigating\researching the Windows NT kernel structure and working principles[95]. Using WRK we could extend the operating system for further research like implementing a new system call[96].

Overall, using the WRK's guidelines we could build the kernel for x84/x64. However, the kernel is not enough but the rest of the Windows OS was not distributed with WRK[97]. Also, WRK includes source for processes, threads, LPC, virtual memory, scheduler, object manager, I/O manager, synchronization, worker threads, kernel heap manager and other core NTOS functionality[98].

Lastly, WRK has been part of the "Windows Academic Program" which supplied universities with concert\code\projects for integrating Windows kernel technologies for teaching\researching. Beside WRK the program includes also CRK (Windows Curriculum Resource Kit) and ProjectOZ experimental environment[99].

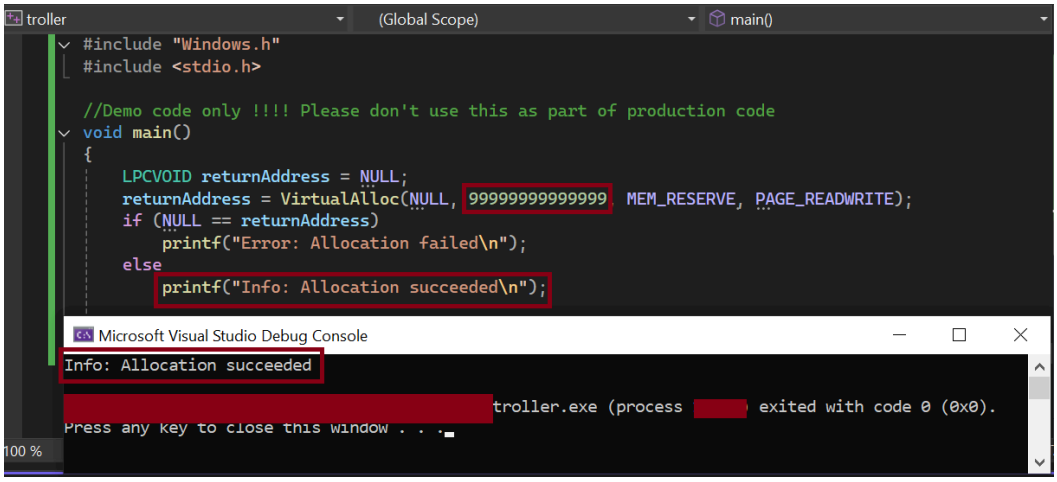
[95] https://betawiki.net/wiki/Windows_Research_Kernel
[96] https://osm.hpi.de/wrk/2007/07/howto-implementation-of-new-system-service-calls/
[97] https://blog.adamfurmanek.pl/2018/07/21/windows-research-kernel-part-1/index.html
[98] https://www.academicresourcecenter.net/curriculum/pfv_ID_7366.html
[99] https://web.archive.org/web/20130624215459/http://www.microsoft.com/education/facultyconnection/articles/articledetails.aspx?cid=2416&c1=en-us&c2=0

Reserved Memory

Memory is allocated in Windows in two stages: “Reserving Memory” and “Committing Memory”. The first just reserves memory which is basically just maintaining the ownership of contiguous memory blocks. We can reserve more memory than what we have in the system due to the fact that reserved memory does not represent real memory in RAM\pagefile - as shown in the screenshot below. Before using those memory blocks we will need to commit them[100].

Overall, we can reserve memory by using Win32 API calls such as “VirtualAlloc” . For reserving memory we need to use the “MEM_RESERVE” flag for allocation type[101]. We can also leverage other API functions such as “VirtualAllocEx”[102] or the “VirtualAlloc2” function[103] .

Lastly, the information if a range of pages are reserved is stored as part of the VAD (Virtual Address Descriptor) of the process - more on that in a future writeup. In case of a memory dump we can extract that information using the “vadinfo” command[104].



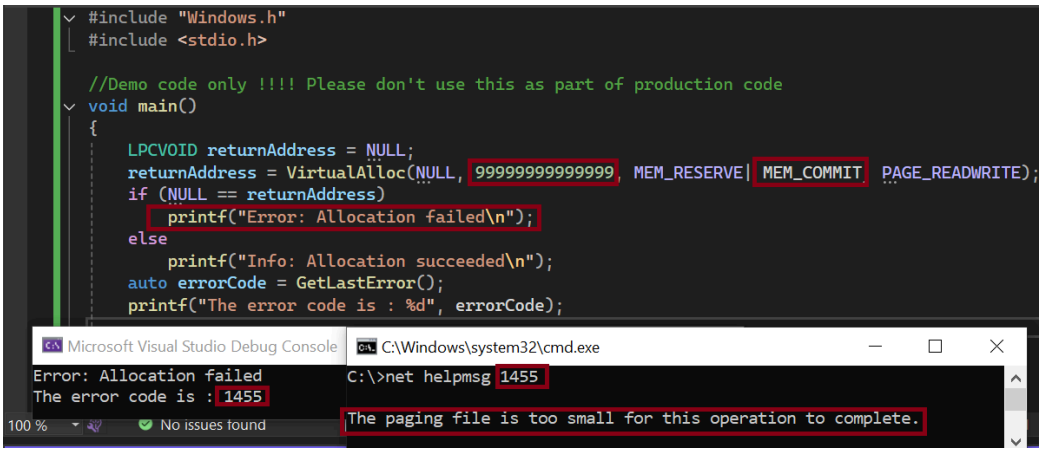
[100] https://answers.microsoft.com/en-us/windows/forum/all/what-is-the-committed-memory-in-process-explorer/616ec11f-7924-4cc3-bdec-1761938f666c
[101] https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc
[102] https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex
[103] https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc2
[104] https://github.com/volatilityfoundation/volatility/wiki/command-reference

Committed Memory

"Committed Memory" is the number of bytes that have been allocated for which the OS has committed a RAM page frame and\or page slot in the page file[105]. Thus, it is the memory which is actually in use and if we access it translates a page to a physical frame or to the pagefile. If we have a committed memory which is not in the working set it is probably in the pagefile[106].

Overall, we can commit memory by using Win32 API calls such as "VirtualAlloc" . For committing memory we need to use the "MEM_COMMIT" flag for allocation type[107] - as shown below. We can also leverage other API functions such as "VirtualAllocEx"[108] or the "VirtualAlloc2" function[109].

Lastly, as opposed to "Reserved Memory"[110] we can't commit more memory than what we have in the system due to the fact that committed memory does represent real memory in RAM\pagefile - as shown in the screenshot below



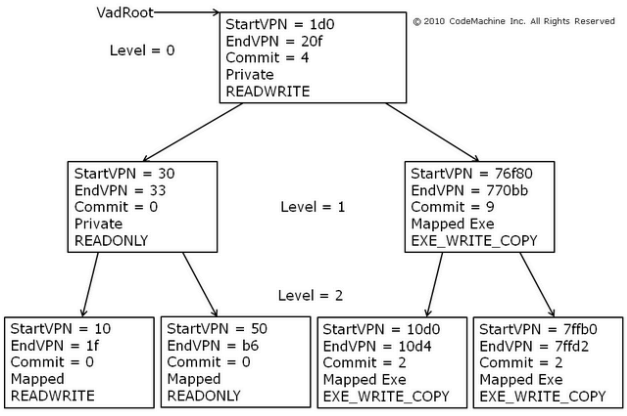
[105] https://answers.microsoft.com/en-us/windows/forum/all/what-is-the-committed-memory-in-process-explorer/616ec11f-7924-4cc3-bdec-1761938f666c
[106] https://www.linkedin.com/pulse/delving-deep-windows-memory-management-brahim-redouane-mellah--uzaee
[107] https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc
[108] https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualallocex
[109] https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-virtualalloc2
[110] https://medium.com/@boutnaru/the-windows-concept-journey-reserved-memory-6a411a0cea3d

VAD Tree (Virtual Address Descriptor Tree)

VAD Trees (Virtual Address Descriptor Trees) are used by the Windows operating system in order to describe memory ranges used by a process as they are allocated (by the memory manager). When a process allocates memory (like by using VirtualAlloc) an entry is created in the VAD tree. However, the corresponding page directory and/or page table entries are not created until a reference to the allocated memory is made. Thus, it can provide significant memory savings[111].

Overall, the VAD structure is a self-balancing tree with child nodes on the left and right side of the parent node. They are AVL (Adelson-Velsky and Landis) self-balancing binary search trees[112]. Each node in the VAD tree has flags associated with it, such as: protection (type of access allowed to the memory region like read, write and execute) and private memory (committed regions that cannot be shared with other processes)[113] - an example is shown in the diagram below[114].

Lastly, when using WinDbg we can leverage the "!vad" extension in order to display the details of a specific VAD or tree of VADs[115]. Also, we can reload the user-mode modules for a specified process by using the VADs of that process in WinDbg by using "!vad_reload"[116].



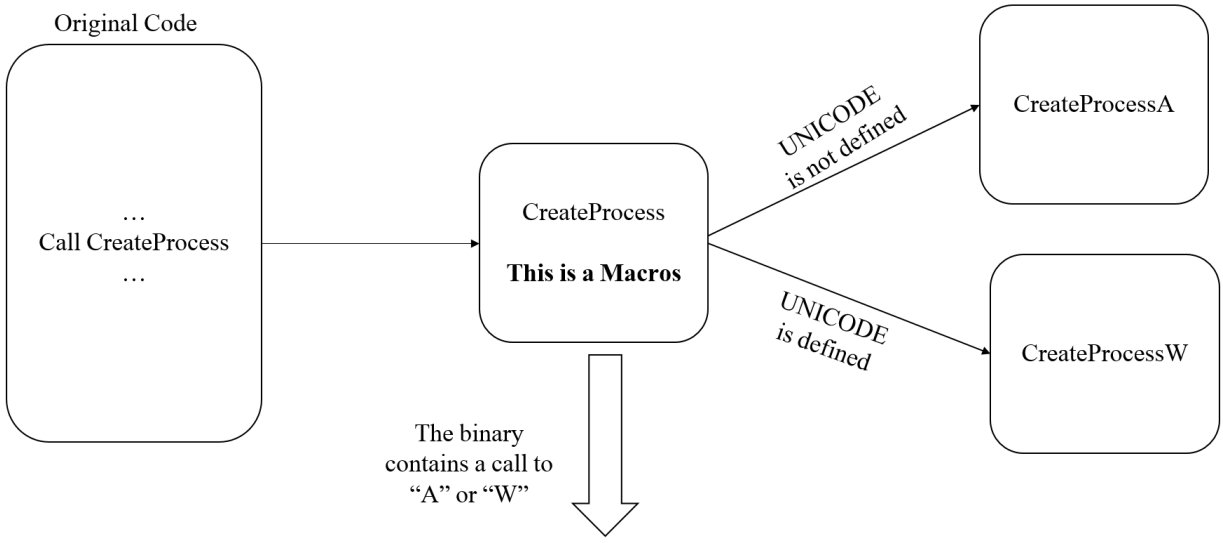
[111] https://www.sciencedirect.com/science/article/pii/S1742287607000503
[112] https://stackoverflow.com/questions/20772834/memory-analysis-vad-tags-and-code-injection
[113] https://www.infosecinstitute.com/resources/penetration-testing/finding-enumerating-processes-within-memory-part-2/
[114] https://www.tophertimzen.com/resources/cs407/slides/week03_01-MemoryInternals.html
[115] https://learn.microsoft.com/en-us/windows-hardware/drivers/debuggercmds/-vad
[116] https://learn.microsoft.com/en-us/windows-hardware/drivers/debuggercmds/-vad-reload

Win32 API - Working with Strings

After Microsoft added support for Unicode as part of Windows, it still needed to support the usage of ANSI strings. The way Microsoft decided to do it is by providing two sets of API (one for ANSI and the second for Unicode. It is important to know that the ANSI version of the API converts the strings to Unicode before calling the relevant syscalls (the kernel is Unicode only).

For example, in order to create a new process we can use “CreateProcessA” or “CreateProcessW” (A for ANSI strings and W for wide char aka Unicode). You might remember that you called CreateProcess and none of the above - so how did it work? The trick is using macros. You called a macro that checked if the UNICODE was defined. If it was defined it made a call to the function ending with “W” else it called the one ending with “A” - see the illustration below for the entire flow.

By the way, in the documentation (like MSDN) the functions names are without the suffix (“A” or “W”) despite the fact it is the name of the macro and not of the functions themself. After compilation the executable contains a reference/dependency to a specific function (you can see it in the diagram below - I have extracted the strings from different DLLs showing the symbol names).




```
C:\Windows\System32>C:\tools\strings64.exe *.dll | findstr /i CreateProcess
C:\Windows\System32\aadtb.dll: DesktopAPI::CallCreateProcess
C:\Windows\System32\aadtb.dll: CreateProcessW
C:\Windows\System32\accessibilitycpl.dll: CreateProcessW
```

Window Messages

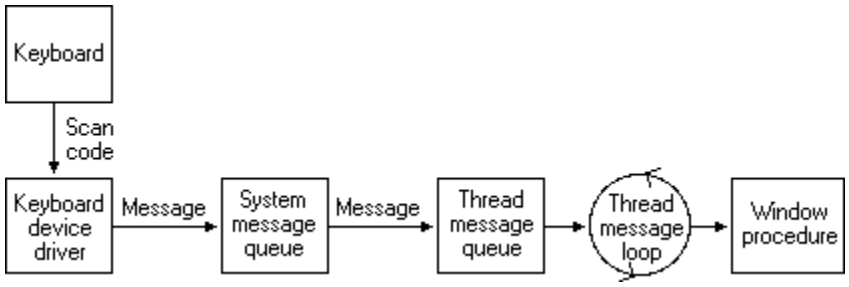
GUI applications under Windows react to different events from the user and the operating system itself. In case of user events, think about: keys pressed, touch-screen gestures, mouse clicks and more. In the case of OS events, think about plugging a new hardware device or changing a power-state[117] (like hibernation or sleep).

Overall, Windows communicates with windows created by applications using messages (aka "Window Messages". In essence, a message is a number that defines a specific event sent to a window. Every window has a "Window Procedure", which is a function that processes all messages sent to windows of the same class[118].

By the way, from the perspective of the Windows kernel the following classes can be created using the "CreateWindow" API call: BUTTON, COMBOBOX, EDIT, LISTBOX, MDICLIENT, SCROLLBAR and STATIC[119].

Moreover, due to the fact an application can receive many messages and it can have several windows (each with its own window procedure) a loop to retrieve the message is needed. It is called "The Message Loop" which dispatches the message to the correct window[120].

To summarize, in a Windows GUI application, "The Message Loop" is a continuous loop that retrieves messages from the operating system and dispatches them to the appropriate "Window Procedure". It is responsible for handling all of the user input (and other events) that occur in the application - as shown in the diagram below[121]. In order to pull a message from the queue we can call "GetMessage"[122].



[117] https://learn.microsoft.com/en-us/windows/win32/learnwin32/window-messages
[118] https://learn.microsoft.com/en-us/windows/win32/winmsg/window-procedures
[119] http://winapi.freetechsecrets.com/win32/WIN32CreateWindow.htm
[120] https://learn.microsoft.com/en-us/windows/win32/learnwin32/window-messages
[121] https://learn.microsoft.com/en-us/windows/win32/inputdev/about-keyboard-input
[122] https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-getmessage

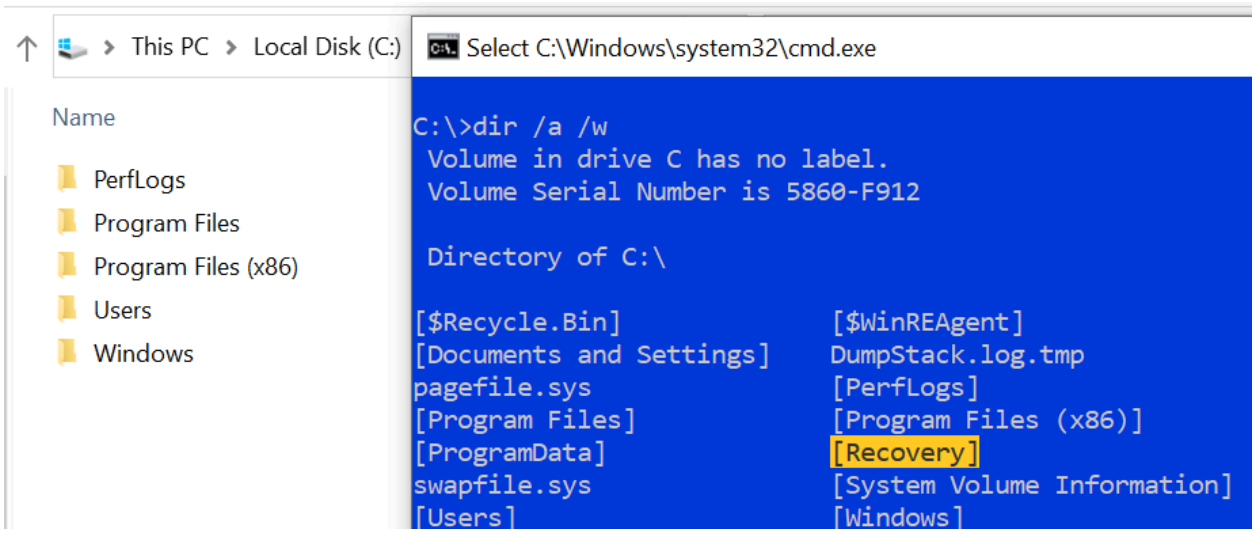
Recovery Directory

A bunch of folks have asked me about what is the goal of different directories in a Windows filesystem hierarchy. So I have decided to write a short series about that. In this writeup we are going to talk about the "Recovery" directory.

It could be that you have never seen this directory before on your root drive ("C:\Recovery"). The reason for that is that the directory is marked "hidden" - as shown in the screenshot below.

By the way it is not enough to display hidden items in explorer to see it. In order to show it we need to unmark "Hide Protected Operating system files (recommended)"[123] - you can see the entire flow in the following link.

Overall, the directory is a leftover from a previous version of Windows (the version before an upgrade that was made). It is used in cases where there are issues after an upgrade and the user wants to revert back. Thus, after a successful upgrade you can probably delete it[124].



[123] https://www.techbout.com/hidden-system-files-windows-10-51145/
[124] https://answers.microsoft.com/en-us/windows/forum/all/can-you-remove-the-large-crecovery-folder-in/e2185d3b-930c-41c0-a1d5-62c753b8a085

COM (Component Object Model)

COM (Component Object Model) is a platform-independent, distributed, object-oriented system for creating binary software components that can interact with each other. COM is the foundation technology for Microsoft's OLE (compound documents) and ActiveX (Internet-enabled components) technologies. These objects can be within a single process, in other processes, even on remote computers[125].

COM was introduced by Microsoft in 1993. It is used for IPC (Inter Process Communication) in a variety of programming languages. Also, COM allows the reuse of objects without any knowledge of their internal implementation, it forces component implementers to provide well-defined interfaces that are separated from the implementation[126].

Let us go over a small example of using COM. Excel uses COM to enable users to create/modify/save/share excel files. By using COM we don't need to understand the binary format of excel files in order to perform the different operations. You can see a demonstration for that in the screenshot below.

Moreover, COM objects are registered with the operating system so they could be loaded in the future. The magic behind that is CLSID (Class ID). A CLSID is a globally unique identifier that identifies a COM class object. If your server or container allows linking to its embedded objects, you need to register a CLSID for each supported class of objects[127].

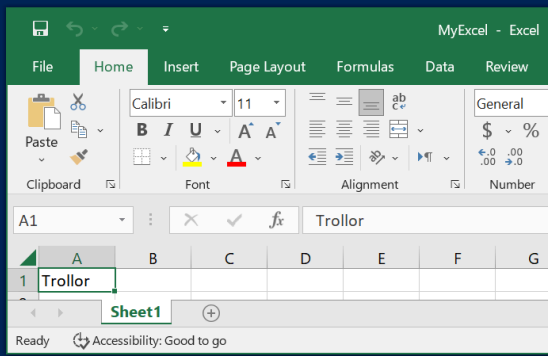
CLSID is stored in the registry under HKEY_CLASSES_ROOT\CLSID\{CLSID value}[128]. It is used by the operating system to locate the appropriate code for loading. For examples of CLSIDs I suggest going over the following link https://www.elevenforum.com/t/list-of-windows-11-clsid-key-guid-shortcuts.1075/.

They are several related technologies that we are going to talk about in future writeups: COM+, DCOM, Windows Runtime (aka WinRT), XPCOM (aka nano-COM), .NET framework, DEC/RPC, OLE, ActiveX, MSRPC and DDE[129]

[125] https://learn.microsoft.com/en-us/windows/win32/com/component-object-model--com--portal
[126] https://en.wikipedia.org/wiki/Component_Object_Model
[127] https://learn.microsoft.com/en-us/windows/win32/com/clsid-key-hklm
[128] https://www.trendmicro.com/vinfo/us/security/definition/clsid
[129] https://learn.microsoft.com/en-us/windows/win32/com/component-object-model--com--portal

Windows PowerShell

```
PS C:\tmp> dir


    Directory: C:\tmp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          1/6/2023  10:47 AM              0 empty.txt


PS C:\tmp> $MyExcelFile = New-Object -ComObject Excel.Application
PS C:\tmp> $MyWorkbook = $MyExcelFile.Workbooks.Add()
PS C:\tmp> $MySheet = $MyWorkbook.Sheets.Item(1)
PS C:\tmp> $MySheet.Range("A1").Value = "Trollor"
PS C:\tmp> $MyWorkbook.SaveAs("C:\tmp\MyExcel.xlsx")
PS C:\tmp> $MyWorkbook.Close()
PS C:\tmp> $MyExcelFile.Quit()
PS C:\tmp> dir


    Directory: C:\tmp


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----          1/6/2023  10:47 AM              0 empty.txt
-a----          1/6/2023  10:48 AM           8668 MyExcel.xlsx


PS C:\tmp> start .\MyExcel.xlsx
PS C:\tmp>
```

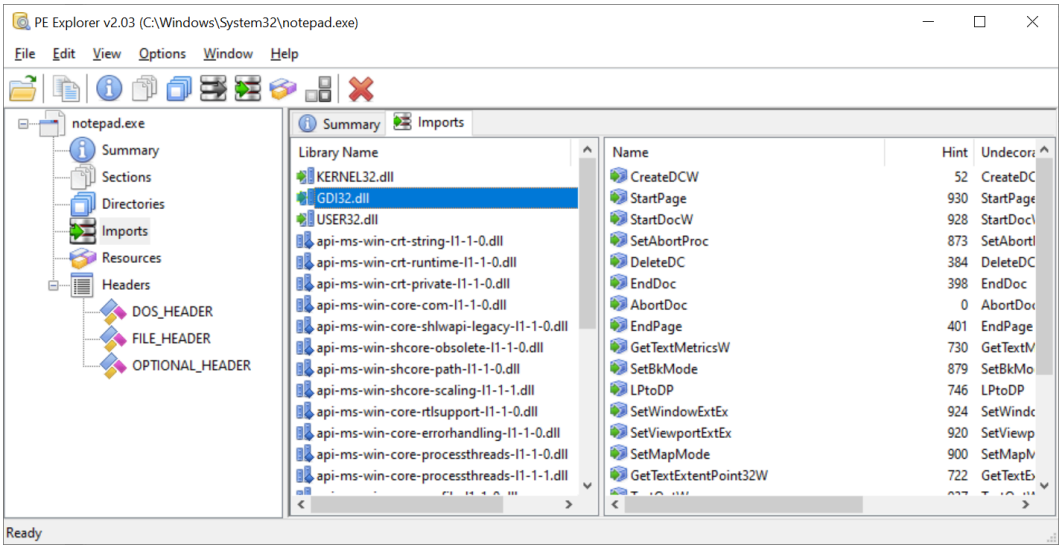


DLL (Dynamic Link Library)

A DLL (Dynamic Link Library) is a PE (Portable Executable) binary which contains data and code (functions) that can be used by other "*.dll" or "*.exe" (applications) files. A DLL can have two types of functions: exported functions and internal functions. The exported functions are meant to be used by other applications or DLLs[130].

Overall, there are two major ways of calling an exported function from a DLL. The first, "Load Time Dynamic Linking" in which a module (exe/dll) calls an exported function as if it was a local function (the relevant DLL is loaded by the kernel). The second, "Run Time Dynamic Linking" in which a programmer needs to call "LoadLibrary"/"LoadLibraryEx" in order to load a DLL at runtime. Also, to get the address of an exported function (from the runtime loaded DLL) we need to use the "GetProcAddress" function[131].

Lastly, in case of "Load Time Dynamic Linking" the PE file contains metadata containing the required DLLs the operating system should load and the names of the used functions/variables from the loaded DLLs - as shown in the screenshot below, which was taken by using "PE Explorer"[132]. By the way, in case of a "Run Time Dynamic Linking" the mentioned metadata is not included.



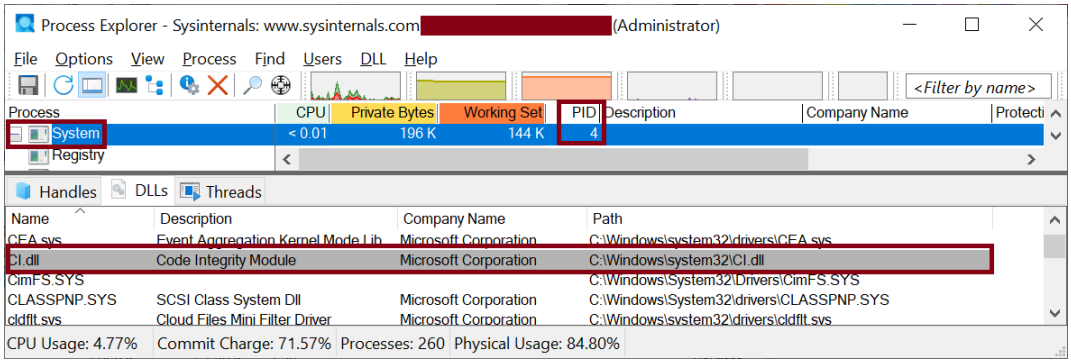
[130] https://learn.microsoft.com/en-us/windows/win32/dlls/dynamic-link-libraries
[131] https://learn.microsoft.com/en-us/windows/win32/dlls/about-dynamic-link-libraries
[132] https://github.com/zodiacon/PEExplorerV2

Not All DLLs Are Loaded\Mapped in User-Mode

“Dynamic Link Library” which are PE files are mostly known for being used in user mode[133]. However, there are also “*.dll” file which are loaded\Mapped into the memory address space of the kernel of the operating system. We can see that by viewing the “DLLs” tab of the “System” process (PID 4) using “Process Explorer”[134] - as shown in the screenshot below.

Overall, DLLs which are loaded into user-mode processes have the “Windows CUI” subsystem defined as part of their PE header. On the other hand, DLLs loaded to are configured with a “Native” subsystem, which is the same as with the “smss.exe”[135] user-mode process.

Lastly, examples of such DLLs (which are loaded to the kernel address space) are: “ci.dll” (Code Integrity Module), “BOOTVID.dll” (VGA Boot Driver) and “cdd.dll” (Canonical Display Driver). As with kernel drivers (“*.sys” files) those DLLs also import functions from “%windir%\system32\ntoskrnl.exe”[136].



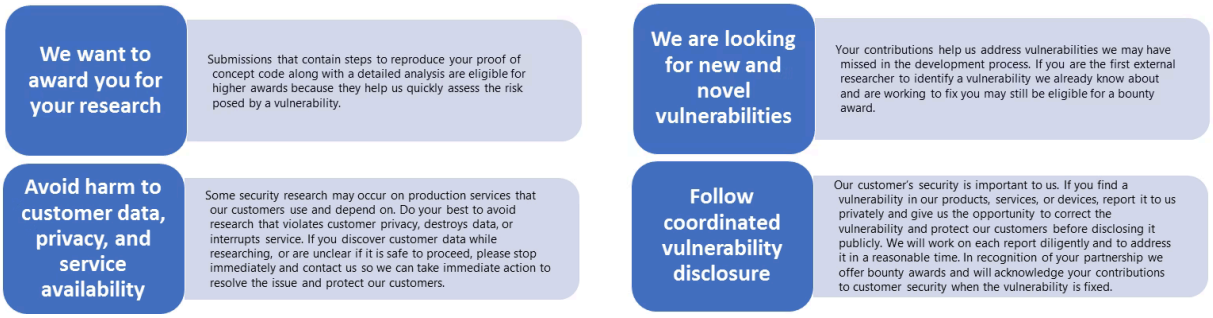
[133] https://medium.com/@boutnaru/the-windows-concept-journey-dll-dynamic-link-library-dceed545416d
[134] https://learn.microsoft.com/en-us/sysinternals/downloads/process-explorer
[135] https://medium.com/@boutnaru/the-windows-process-journey-smss-exe-session-manager-subsystem-bca2cf748d33
[136] https://connormcgarr.github.io/Kernel-Exploitation-2/

MSRC (Microsoft Security Response Center)

MSRC (Microsoft Security Response Center) is part of the defender community. For more than 20 years MSRC has engaged with security research in order to protect Microsoft's customers[137]. Thus, MSRC's mission is defined as protecting customers and Microsoft from current and emerging threats related to security and privacy[138].

Overall, MSRC also provides a bug bounty program. Among the bounty program we can find: cloud programs ("Microsoft Azure", "Microsoft Identity", "Xbox", "M365", "Microsoft Azure DevOps Services", "Microsoft Dynamics 365 and Power Platform", Microsoft AI"and "Microsoft Defender"), platform programs ("Microsoft Hyper-V", "Microsoft Windows Insider Preview", "Microsoft Applications and On-Premises Servers", "Windows Defender Application Guard", "Microsoft Edge (Chromium-based)" and "Microsoft 365 Insider") and defense & grant programs ("Mitigation Bypass and Bounty for Defense", "Grant: Microsoft Identity" and "SIKE Cryptographic Challenge"). Each program has its own scope, however they share the same high level requirements - as shown below[139].

Lastly, MSRC investigates all reports of security vulnerabilities affecting Microsoft products and services. Also, MSRC provides the information here as part of the ongoing effort to help manage security risks and help keep systems protected[140]. By the way, there is a GitHub repository which stores security research conducted by MSRC[141].



[137] https://www.microsoft.com/en-us/msrc
[138] https://www.microsoft.com/en-us/msrc/mission?rtc=1
[139] https://www.microsoft.com/en-us/msrc/bounty
[140] https://msrc.microsoft.com/update-guide/en-us
[141] https://github.com/microsoft/MSRC-Security-Research

Windows Services

Before we are going to talk about processes which are related to services handled under Windows (like services.exe and svchost.exe) we have to explain what a service is.

Services are processes which are managed by the operating system, it resembles demons in Linux (but there are a couple of differences that I am going to talk about in a different writeup).

Due to security reasons services can be executed at least under 3 different entities: System, Network Service and Local Service (each of them with different permissions and privileges - we will cover them in more details in the future). Of course, we can also run a service using a local user or a domain user with any access rights that we want.

There are different ways in which we can administer services, however I am going to focus on probably the 4 well known interfaces. First, Win32 API (it is used also for 64 bit despite its name) such as StartService[142]. Second, the mmc snap-in "services.msc"[143]. Third, PowerShell by leveraging cmdlets such as: New-Service, Get-Service, Restart-Service, Stat-Service, Stop-Service and more. Fourth, the command line tool "sc.exe"[144].

A service can be in one of the three major states: started, stopped or paused. Alos, each service has a startup state which defines what should happen with the service when the OS starts - it could be one of the following: Automatic, Automatic (Delayed Start), Manual or Disabled. Let us go over each and one of them.

Automatic, in this configuration the SCM starts the service as part of the system boot process.
In the case of the delayed start, it's an optimization feature to reduce the time it takes the system to boot-up. "Automatic Start" is still run by the SCM but not during the boot process (they are started automatically shortly after the boot process has finished). Manual, in this configuration the SCM does not start the service and it needs to be from some other administrative interface (as we explained above), we can also script it if we want. Disabled, in this configuration even an administrator can't start the service. In order to start the service we first need to enable it by setting it to any setting which is not disabled.

Over the years different security enhancements were added for service hardening (Examples are session isolation, least privileges, restricted network access and service isolation). We are going to speak about it more when talking about security and the process of hardening the OS.

---

[142] https://docs.microsoft.com/en-us/windows/win32/services/service-functions
[143] https://www.thewindowsclub.com/open-windows-services
[144] https://ss64.com/nt/sc.html

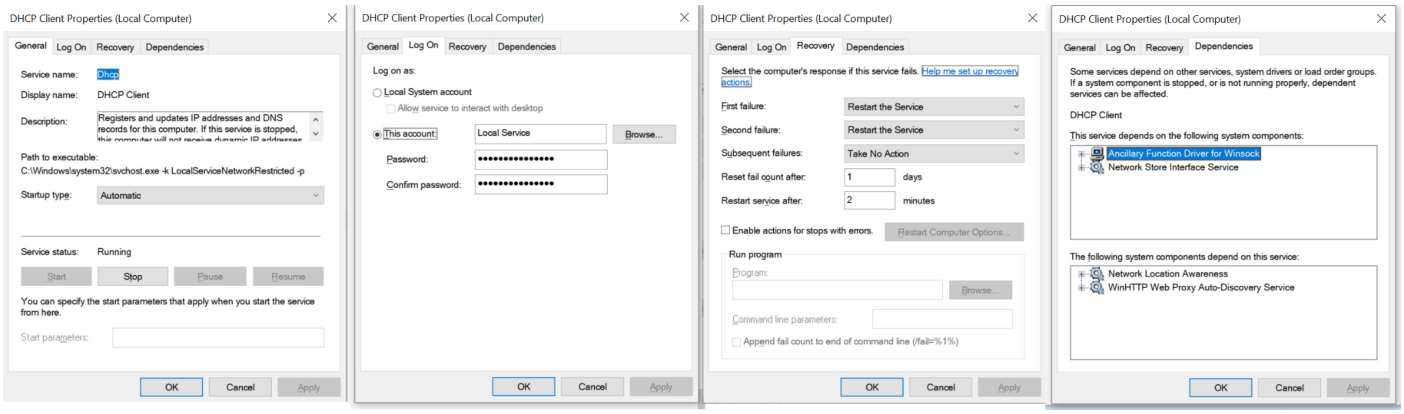
In the screenshot below you can see examples for the things we have talked about regarding the DHCP Client service. On the left we can see the status and the startup type and on the right the user which the service is logging on behalf of.
In addition, I am going to talk about: dependency management and recovery handling. In the screenshot below an example of those configurations are shown regarding the “DHCP Client” service (from services.msc). It is time to deep dive into it, so let’s go.

In the case of dependency management in the configuration of each service there is a list of dependent services and the list of the services that depend on it. Those lists are checked by the SCM when starting and stopping services.

Regarding recovery, the configuration allows setting actions for first/second/subsequent failures. The action could be to restart the service/the computer or execute an arbitrary command. We could also reset the fail count after N (we can set it) days and enable actions for cases the service stops with errors.

All the configurations of services are saved as part of the Registry (HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services). It is important to know that kernel drivers configuration is also stored there (we will go over it as part of the SCM discussion).

Next we are going to check out the SCM (Service Control Manager) which is the OS part responsible for managing all the services and their configuration.



What IPC (Inter Process Communication) mechanisms do we have in Windows?

Due to the fact that each process in Windows has its memory address space[145] we can't pass pointers between threads in different processes and expect to see the same data in the same virtual address. It could be that the virtual address is not valid in one of the address spaces or we have a different data stored there (they are of other cases also).

In order to allow different threads (in different processes) to pass data between them we need to use an IPC (Inter Process Communication) mechanism - an illustration of that is seen in the diagram below[146]. Moreover, each OS has its own IPC mechanisms. On Windows we have the following mechanisms: clipboard, Windows Messages, COM (Component Object Model), DDE (Dynamic Data Exchange), Shared Memory, File Mapping, Mailslots, Pipes, RPC (Remote Procedure Call), ALPC (Advance Local Procedure Call) and sockets[147].

There are also folks that say we can use files on the filesystem for IPC but it is not a specific mechanism of Windows so I won't speak about it for now. Lastly, when talking about IPC we should also talk about synchronization objects, however I will leave it for a future discussion.

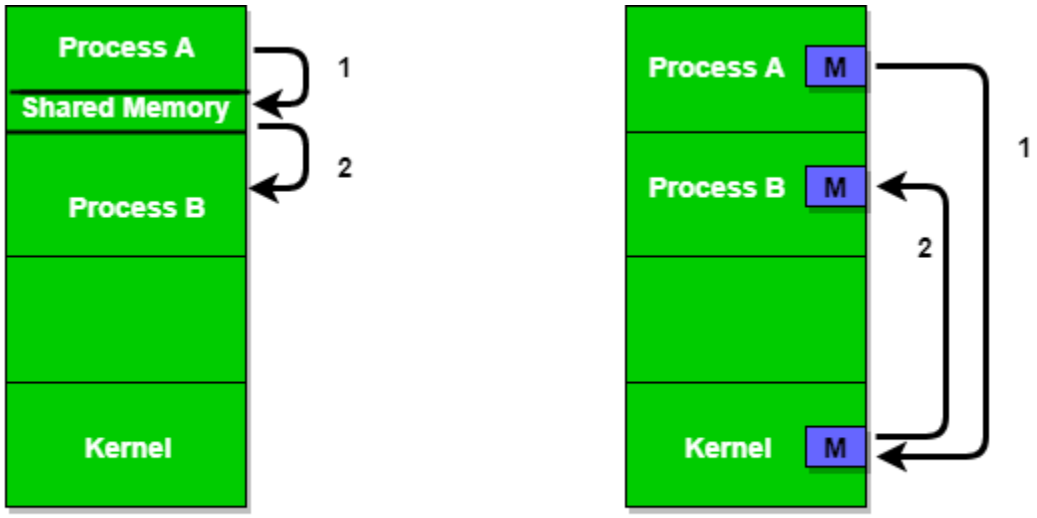


Figure 1 - Shared Memory and Message Passing

[145] https://medium.com/@boutnaru/linux-memory-management-part-1-introduction-896f376d3713
[146] https://www.geeksforgeeks.org/inter-process-communication-ipc/
[147] https://www.slideshare.net/mfsi_vinothr/ipc-mechanisms-in-windows

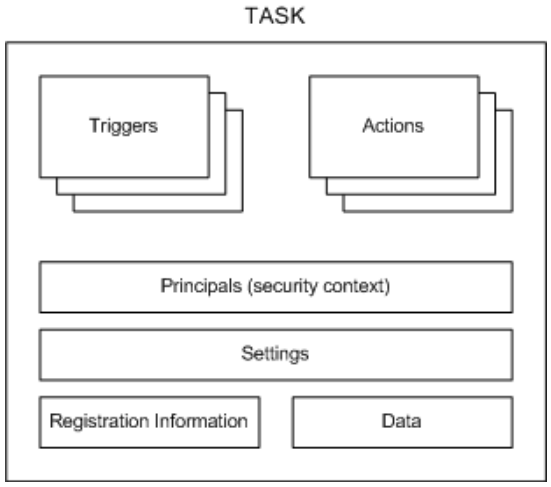
Tasks (Windows Scheduler)

Overall, a task is a scheduled work that is performed by the "Task Scheduler" service. Each task has several components: triggers, actions, principals, settings, registration information and data - as shown in the diagram below[148].

Triggers are events/time-based conditions which are used as a criteria for starting an execution of a task. A task can have multiple triggers up to a maximum of 48[149]. Also, actions are the actual work performed by a task. A task can have a single/multiple actions up to a maximum of 32 actions. We can different types of actions: "ComHandler" (COM), "Exec Action", "Email Action" (sending an email notification) and "Show Message Action"[150].

Moreover, principles is the definition of the security context in which the task is executing on behalf of, including UAC settings and more[151]. Settings, that is the configuration used by the "Task Scheduler" while running the task. Think about if we can run multiple instances of the task, or what to do with the task if the system is in idle state and more. By default a task will stop after 72 hours, unless we change the "ExecutionTimeLimit"[152].

In addition, registration information is the data collected when the task is created/registered. Data elements that can be included (but not limited to) are: author, date, description, task version, security descriptor and more[153]. We can also have additional documentation for the tasks (this is the data portion in the diagram shown below). Lastly, "Task Scheduler" has two versions ("1.0" and "2.0") which have differences in the API they support and the configuration that can be made.



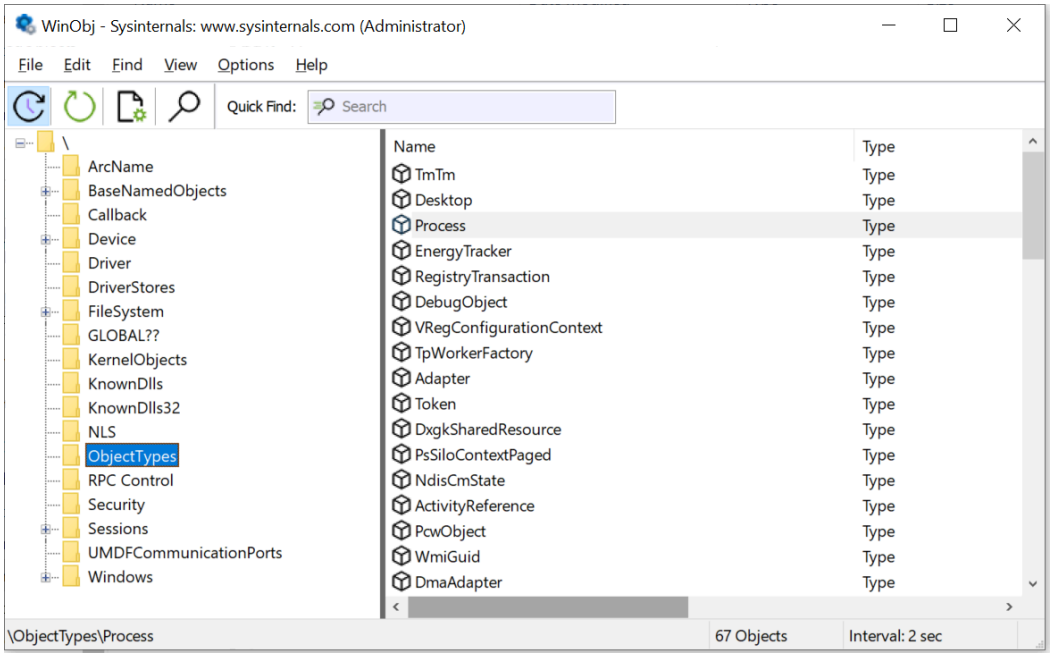
[148] https://learn.microsoft.com/en-us/windows/win32/taskschd/tasks
[149] https://learn.microsoft.com/en-us/windows/win32/taskschd/task-triggers
[150] https://learn.microsoft.com/en-us/windows/win32/taskschd/task-actions
[151] https://learn.microsoft.com/en-us/windows/win32/taskschd/security-contexts-for-running-tasks
[152] https://learn.microsoft.com/en-us/windows/win32/api/taskschd/nf-taskschd-itasksettings-get_executiontimelimit
[153] https://learn.microsoft.com/en-us/windows/win32/taskschd/task-registration-information

Objects

Basically, objects are data structures that represent system resources. Think about things like processes and threads[154]. Those object are divided to two main parts: the object header "struct _OBJECT_HEADER"[155] and the body which holds the specific information regarding a system resource. Examples are: _EPROCESS[156] , _FILE_OBJECT[157] and more.

Moreover, we can see a list of available object types using the "WinObj" tool from Sysinternals - as shown in the screenshot below. By the way, the subsystem that manages the Windows resources is the "Object Manager", which is part of the "Windows Executive". The "Windows Executive" is contained as part of "ntoskrnl.exe"[158].

Overall, they are three different categories of objects in Windows: user, graphics (GDI objects) and kernel[159]. In User we have objects like: "Hook"[160] and "Menu"[161]. In GDI we have objects like: "Font"[162] and "Region"[163]. Lastly, in the case of kernel objects we have examples like: "Desktop"[164] and "Job"[165].



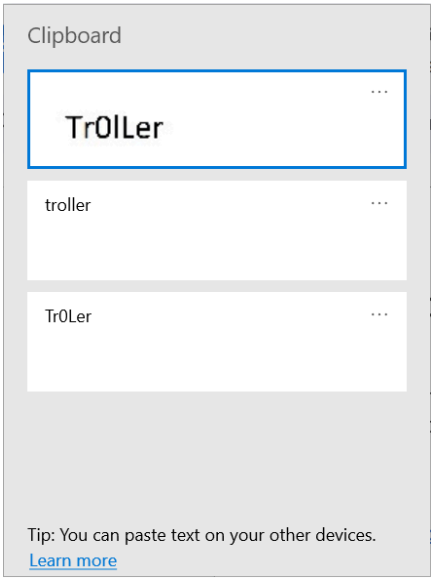
[154] https://learn.microsoft.com/en-us/windows/win32/sysinfo/handles-and-objects
[155] https://www.nirsoft.net/kernel_struct/vista/OBJECT_HEADER.html
[156] https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/ntos/ps/eprocess/index.htm
[157] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/ns-wdm-_file_object
[158] https://learn.microsoft.com/en-us/previous-versions//cc768129(v=technet.10)
[159] https://learn.microsoft.com/en-us/windows/win32/sysinfo/object-categories
[160] https://learn.microsoft.com/en-us/windows/win32/winmsg/hooks
[161] https://learn.microsoft.com/en-us/windows/win32/menurc/menus
[162] https://learn.microsoft.com/en-us/windows/win32/gdi/fonts-and-text
[163] https://learn.microsoft.com/en-us/windows/win32/gdi/regions
[164] https://learn.microsoft.com/en-us/windows/win32/winstation/desktops
[165] https://learn.microsoft.com/en-us/windows/win32/procthread/job-objects

Clipboard

Overall, the clipboard is a temporary storage in Windows that can store different data types. Among those data types are images and text. The operation which stores data in the clipboard is called “copy” (Ctrl+C), also we can use “cut” using “Ctrl+X”. In order to paste data from the clipboard we use “Ctrl+V”[166].

In Windows 10 we can store in the clipboard up to 25 items. We can access the “Clipboard History” using the keyboard shortcut “Winkey + V”. While pressing this shortcut a menu showing the current clipboard history - as shown in the screenshot below[167]. We can see that the first item is an image and the other two are strings.

Lastly, in Windows 10/11 we can also copy text/images from one PC to another using a cloud based clipboard. We can also pin items that we would like to use. In order to do that we need to sign with a Microsoft account/work account[168].



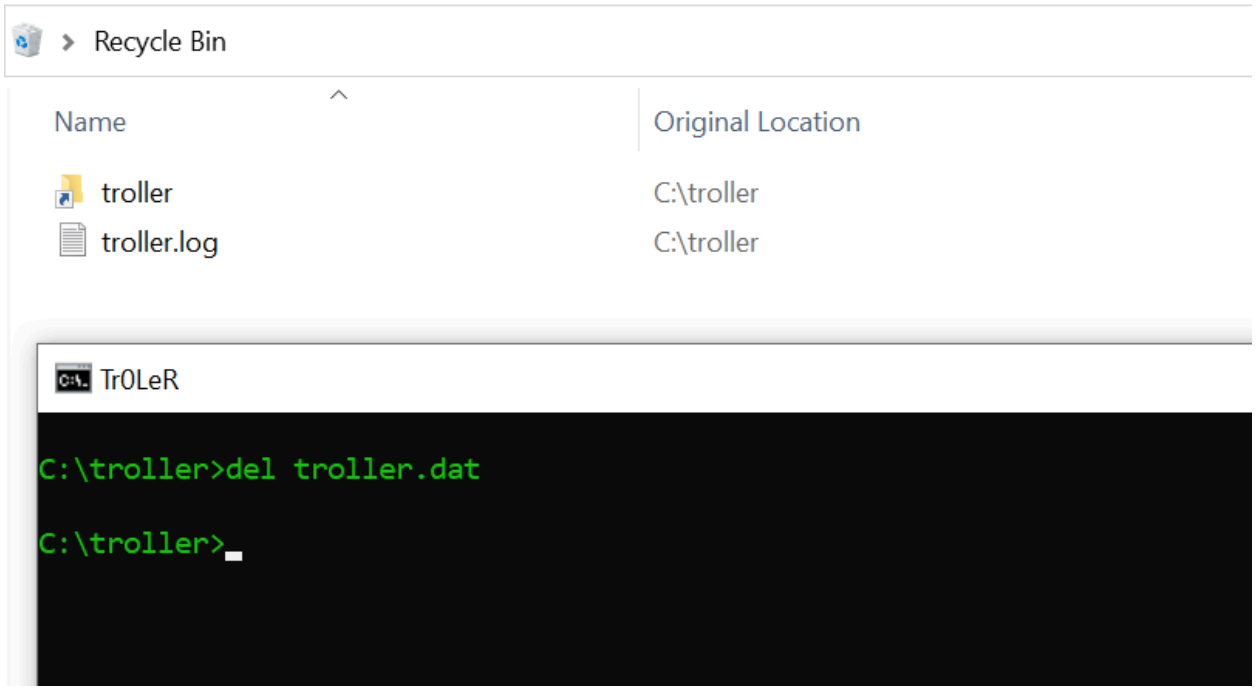
[166] https://www.howtogeek.com/671222/how-to-enable-and-use-clipboard-history-on-windows-10/
[167] https://www.emailoverloadsolutions.com/blog/windows-clipboard-history-feature
[168] https://support.microsoft.com/en-us/windows/clipboard-in-windows-c436501e-985d-1c8d-97ea-fe46ddf338c6

Recycle Bin

The goal of the “Recycle Bin” is to provide a second chance for recovering files/directories that have been deleted. There is a nice trick to get into the “Recycle Bin” by using “cmd /c shell:RecycleBinFolder”. Of course we can just find a shortcut for it on the desktop[169].

Thus, we can say “Recycle Bin” is a folder where deleted items are stored temporarily. We can use the “Recycle Bin” to restore the files to their original location. However, it was not created in a manner in which we can use the files directly from the “Recycle Bin”[170].

Moreover, files are moved to the “Recycle Bin” only if we delete them for “File Explorer” and not when doing it directly from other applications like “cmd.exe”[171] - as shown in the screenshot below. By the way, holding the “Shift” key while deleting a file in “File Explorer” causes it not to be moved to the “Recycle Bin”. Lastly, we can go over a reference implementation of the “Recycle Bin” as part of ReactOS[172].



[169] https://www.digitalcitizen.life/where-is-recycle-bin/
[170] https://www.techopedia.com/definition/4675/recycle-bin
[171] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b
[172] https://github.com/reactos/reactos/tree/master/dll/win32/shell32/shellrecyclebin

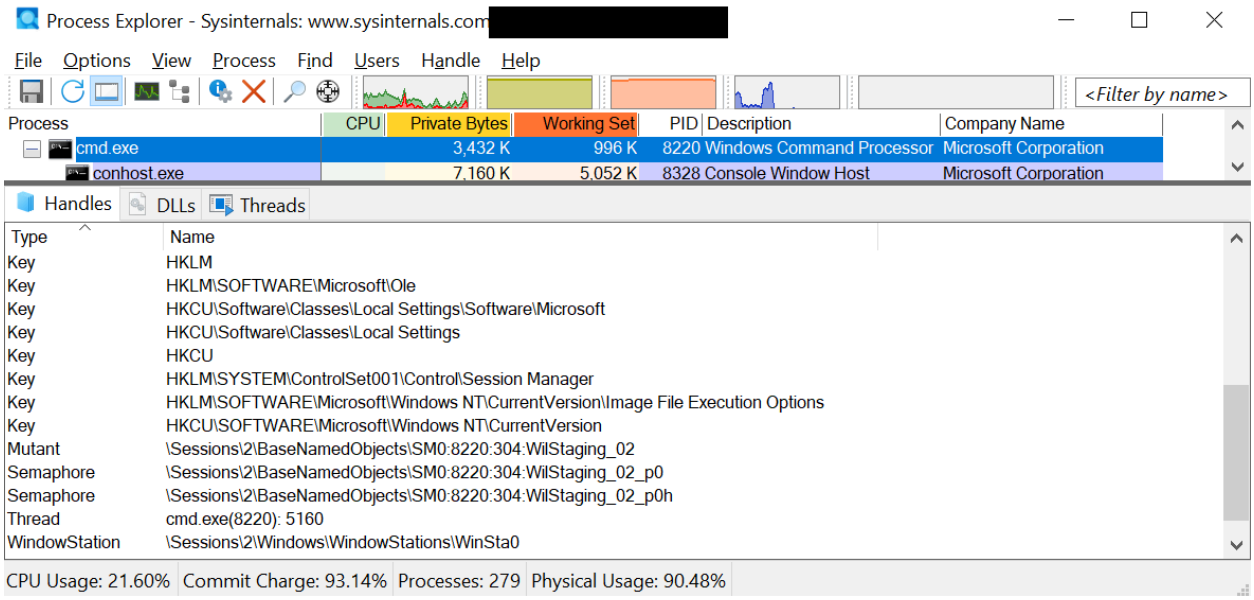
Handles

Overall, applications running in user mode can't directly access the Windows objects[173] or object data held by the "Object Manager". Due to that, an application needs to obtain a handle to the specific object. For each handle there is an entry in a handle table which resides in kernel space, there is one for each process[174].

Thus, an object in Windows is accessed by the user using a per process "handle table". Opening an object results in adding a pointer to the object to the specific "handle table". The return value is "an index" to that table[175].

We can see when using different Win32 API function like when opening/creating a file using "CreateFileW" the return value is of type HANDLE[176].

Lastly, we can use Sysinternals' tools like "Process Explorer" and handle.exe[177] in order to see which open handles each process has. An example showing the handles for "cmd.exe" using "Process Explorer" is shown in the screenshot below.



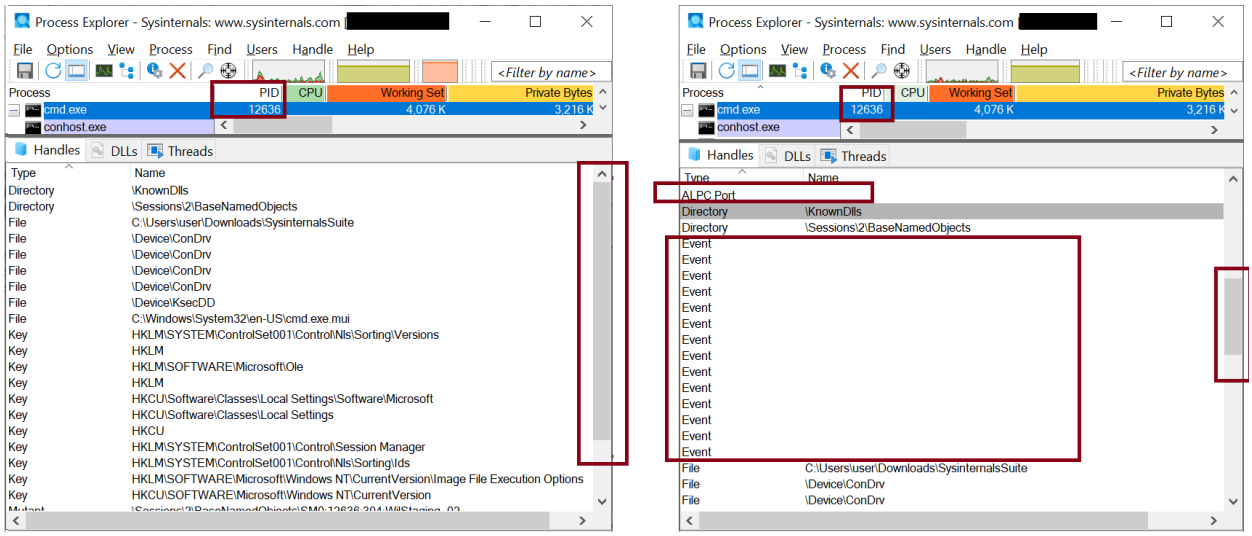
[173] https://medium.com/@boutnaru/windows-objects-2c289da600bf
[174] https://learn.microsoft.com/en-us/windows/win32/sysinfo/handles-and-objects
[175] https://www.cs.miami.edu/home/burt/journal/NT/handle_table.html
[176] https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilew
[177] https://learn.microsoft.com/en-us/sysinternals/downloads/handle

Unnamed Handles

Windows objects[178] are inaccessible directly from user-mode code. Handles are used as "indexes" in a per-process handle table (stored in the kernel address space). Thus, when opening an object a new handle entry is created[179]. We are going to focus on handles which don't have a name aka "unnamed handles" (as opposed to named handles).

Overall, we can't treat an unnamed handle as a secure object by default. If an unnamed handle has a NULL DACL as part of its security descriptor[180] it grants access to everyone. However, even if there is no name to a handle it does not mean it can't be accessed - it just means we can't access it using a name. We can still use the "DuplicateHandle" Win32 API function call to duplicate it[181] or inherit the handle[182].

Lastly, we can see unnamed handles using "Process Explorer". However, it is not enabled by default. In order to enable it we can press "View->Show Unnamed Handles and Mappings". Then in the handles tab both named and unnamed handles are displayed - as shown in the screenshot below.



[178] https://medium.com/@boutnaru/windows-objects-2c289da600bf
[179] https://medium.com/@boutnaru/windows-handles-594b36c39d2f
[180] https://medium.com/@boutnaru/the-windows-security-journey-dacl-discretionary-access-control-list-c74545e472ec
[181] https://learn.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-duplicatehandle
[182] https://devblogs.microsoft.com/oldnewthing/20150604-00/?p=45451

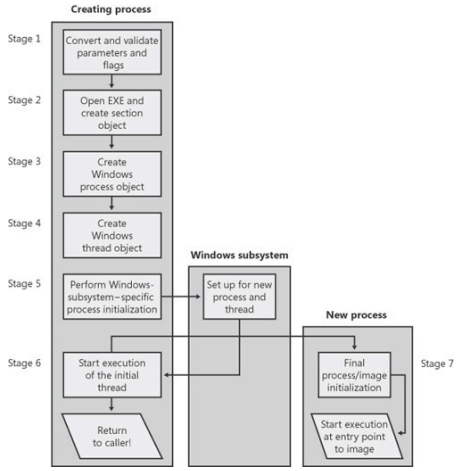
Processes

From the perspective of the Windows operating system an application is composed of one or more processes. Thus, we can say a process is an executing program[183]. Moreover, every process provides the resources needed for the application to run. In order to support that a process in Windows has: a virtual memory address space, executable code, open handle table, unique identifier (PID), environment variables, priority, access token and more attributes[184].

Overall, for creating new processes under Windows we can use various Win32 API calls such as: "CreateProcessA"\"CreateProcessW"[185], "CreateProcessAsUserA"\"CreateProcessAsUserW"[186] and "CreateProcessWithLogonW"[187].

The function "CreateProcessA"\"CreateProcessW" creates the new process with the security context of the calling process. In the case of "CreateProcessAsUserA"\"CreateProcessAsUserW" we can provide an handle to the token[188] we want to grant the new process created. With "CreateProcessWithLogonW", we can provide the username and password for the security context we want to execute the new process to execute with. For reference implementation I suggest going over ReactOS's source code of "NtCreateProcessEx" which calls "PspCreateProcess"[189].

Lastly, a process has at least one execution flow which is a thread (aka the main/primary thread) - more on threads in a separate writeup. Also, I am going to describe the different data structures used by Windows in order to manage processes in future writeups. An overview of the stages Windows follows when creating a process is shown in the diagram below[190].



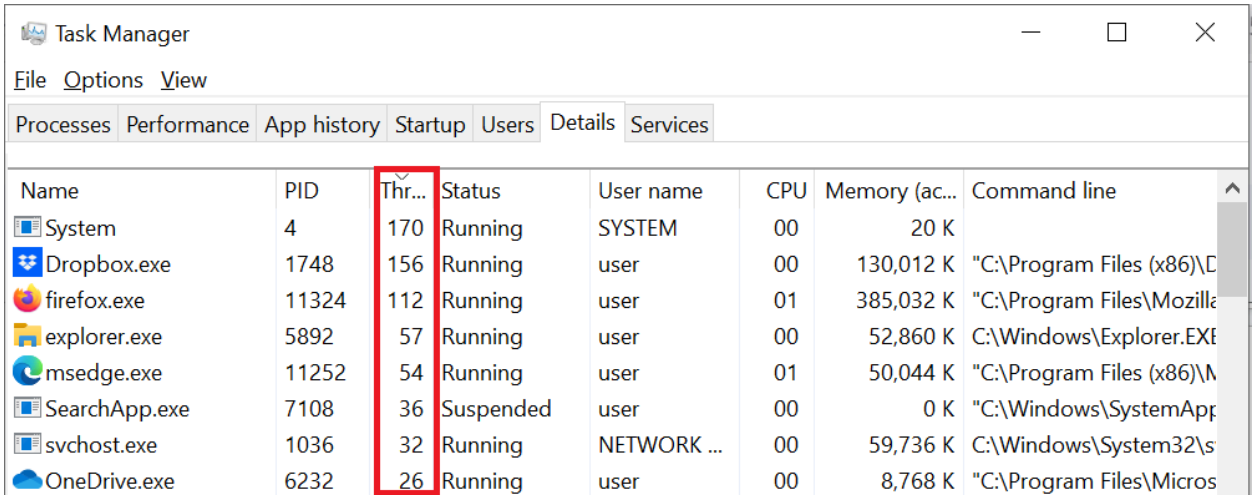
[183] https://learn.microsoft.com/en-us/windows/win32/procthread/processes-and-threads
[184] https://learn.microsoft.com/en-us/windows/win32/procthread/about-processes-and-threads
[185] https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessasuserw
[186] https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessasuserw
[187] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createprocesswithlogonw
[188] (https://medium.com/@boutnaru/windows-security-access-token-81cd00000c64
[189] https://github.com/reactos/reactos/blob/master/ntoskrnl/ps/process.c#L1344
[190] https://www.microsoftpressstore.com/articles/article.aspx?p=2233328&seqNum=3

Threads

In general a Windows process[191] is composed of one or more threads. For the eyes of the operating system, a thread is the basic unit to which CPU time is allocated. A thread can run any code of the process, also the same code path can be executed in parallel by different threads[192]. Each thread has its own identifier (TID, which stands for "Thread ID"). Also, every Windows process has at least one thread - as shown in the screenshot below.

Moreover, in order to create a thread within the virtual memory address space of the calling process we can using the "CreateThread" API call[193]. If we want to create a thread in the virtual address space of a different process we can use "CreateRemoteThread"[194].

Lastly, each thread has its own stack area (both in user mode and in kernel mode) so each one of them gets its own execution flow. The default stack size in user-mode is 1MB[195]. By the way, TID is also divisible by 4 like PID[196].

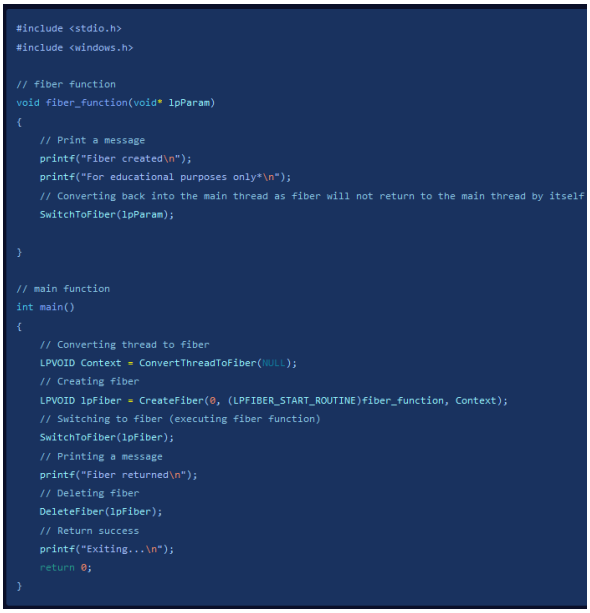


[191] https://medium.com/@boutnaru/windows-process-923b9332c12
[192] https://learn.microsoft.com/en-us/windows/win32/procthread/processes-and-threads
[193] https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createthread
[194] https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createremotethread
[195] https://learn.microsoft.com/en-us/windows/win32/procthread/thread-stack-size
[196] https://medium.com/@boutnaru/windows-why-pids-tids-are-4-divisible-6af16cf4621d

Fiber

Fiber is a unit of execution which is not scheduled by the kernel of Windows, thus it needs to be scheduled by the application itself. Due to that, each fiber executes in the context of the thread[197] that had scheduled it. A code snippet for using fibers as shown in the image below[198]. Overall, we can say that fibers are "lightweight threads" of execution. The big difference between them and OS threads is that they're cooperatively scheduled as opposed to preemptively scheduled. This basically means that fibers yield themself (they withdraw their execution) to allow another fiber to run. Sometimes you can find fibers called: "green threads", "tasklets", "coroutines", "user-space threads" or "microthreads"[199].

Moreover, in order to use fibers we can leverage the Win32 API. "CreateFiber" allows the creation of a new fiber for a thread, the number of fibers per process is limited by the virtual memory[200]. We can use "ConvertThreadToFiber" for converting the current thread into a fiber, which is needed before scheduling other fibers[201]. For both of those function there is an extended version "CreateFiberEx"[202] and "ConvertThreadToFiberEx"[203]. Lastly, for scheduling a fiber we can use the "SwitchToFiber" function[204]. To retrieve the data associated with the current fiber we can use the "GetFiberData" function[205]. When the fiber is not needed anymore just call "DeleteFiber"[206].



```c
#include <stdio.h>
#include <windows.h>

// fiber function
void fiber_function(void* lpParam)
{
    // Print a message
    printf("Fiber created\n");
    printf("For educational purposes only*\n");
    // Converting back into the main thread as fiber will not return to the main thread by itself
    SwitchToFiber(lpParam);

}

// main function
int main()
{
    // Converting thread to fiber
    LPVOID Context = ConvertThreadToFiber(NULL);
    // Creating fiber
    LPVOID lpFiber = CreateFiber(0, (LPFIBER_START_ROUTINE)fiber_function, Context);
    // Switching to fiber (executing fiber function)
    SwitchToFiber(lpFiber);
    // Printing a message
    printf("Fiber returned\n");
    // Deleting fiber
    DeleteFiber(lpFiber);
    // Return success
    printf("Exiting...\n");
    return 0;
}
```

197 https://medium.com/@boutnaru/windows-threads-3a839fa67ae3
198 https://de-engineer.github.io/Processes-threads-jobs-fibers/#creating-a-fiber
199 https://graphitemaster.github.io/fibers/
200 https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createfiber
201 https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-convertthreadtofiber
202 https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createfiberex
203 https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-convertthreadtofiberex
204 https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-switchtofiber
205 https://learn.microsoft.com/en-us/windows/win32/api/winnt/nf-winnt-getfiberdata
206 https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-deletefiber

Mailslot

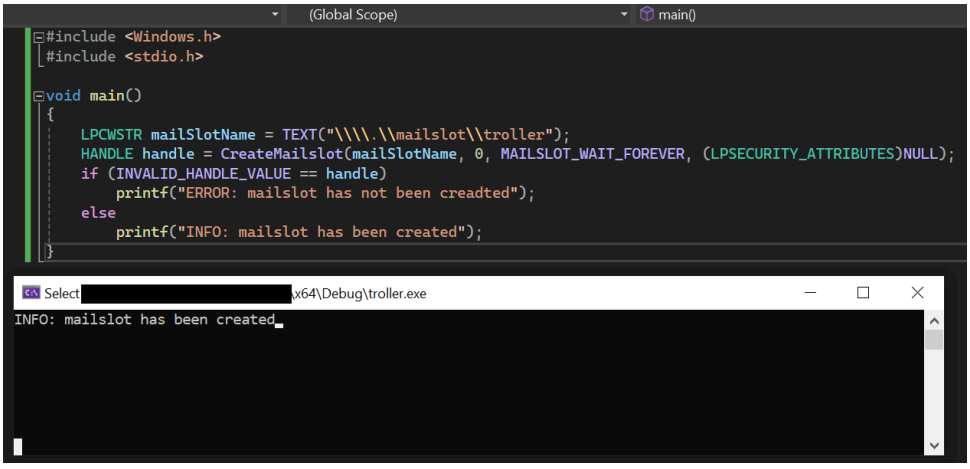
Mailslot is a Windows IPC[207] mechanism. It is used for one-way communication between endpoints. Thus, applications can store data in a mailslot. Those messages can be sent locally or over the network. The owner of the mailslot can retrieve messages stored in the mailslot[208].

Overall, we can think about a mailslot as a "pseudo file" that resides in memory (when the mailslot handle is closed that data is deleted). Due to that we can access it using the files API (like ReadFile/WriteFile/CreateFile). The data stored in a mailslot can be in any format, however it can be larger than 424 bytes when sent between computers[209].

Moreover, when creating a mailslot (by a mailslot server) its name must be in the following pattern "\\.\mailslot\[path\]name". A mailslot client can write messages locally or remotely, when writing data to a remote mailslot we need to use the following "\\ComputerName\mailslot\[path\]name". There is also a way for accessing every mailslot in a domain using "\\DomainName\mailslot\[path\]name" or "*\mailslot\[path\]name" in the system's primary domain[210].

In addition to that, for creating a mailslot we can use the API function "CreateMailslotA"[211] or the wide character version of the API "CreateMailslotW"[212] - as shown in the screenshot below (this is just a toy example, please don't use it "as is" in production code due to security reasons).

Lastly, the remote capability of mailslots is baked on the SMB (Server Message Block) protocol[213]. By the way, Microsoft intends to remove the support of remote mailslots[214].

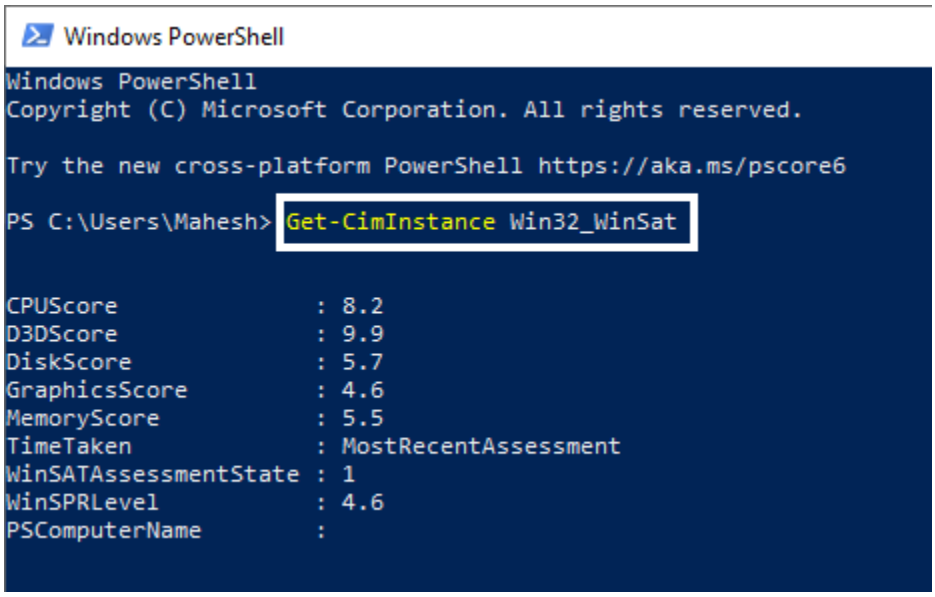


[207] https://medium.com/@boutnaru/windows-ipc-inter-process-communication-introduction-434c9287279b
[208] https://learn.microsoft.com/en-us/windows/win32/ipc/mailslots
[209] https://learn.microsoft.com/en-us/windows/win32/ipc/about-mailslots
[210] https://learn.microsoft.com/en-us/windows/win32/ipc/mailslot-names
[211] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createmailslota
[212] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createmailslotw
[213] https://wiki.wireshark.org/Mailslot.md
[214] https://www.anoopcnair.com/deprecation-of-remote-mailslots-in-windows/

Windows Experience Index (WEI)

Windows Experience Index (WEI) is a scoring mechanism which allows us to understand how fast our computer is and which hardware shortcomings it has. WEI was first introduced as part of Windows Vista. The value of WEI can range from 1.0 (worst performance) to 9.9 (highest performance). It is used by WinSAT (Windows System Assessment Tool), which is based on the lowest score of : RAM, CPU, graphic chipset (or GPU) and hard disk[215].

Overall, for Windows 7 and 8 there was a GUI interface for viewing the WEI score (as part of the control panel). However, since Windows 10 it was removed, we can still calculate the score using "winsat.exe"[216]. We can also use the WMI (Windows Management Instrumentation) class "Win32_WinSAT", which summarizes the information of the recent assessment[217] - as shown in the screenshot below[218].



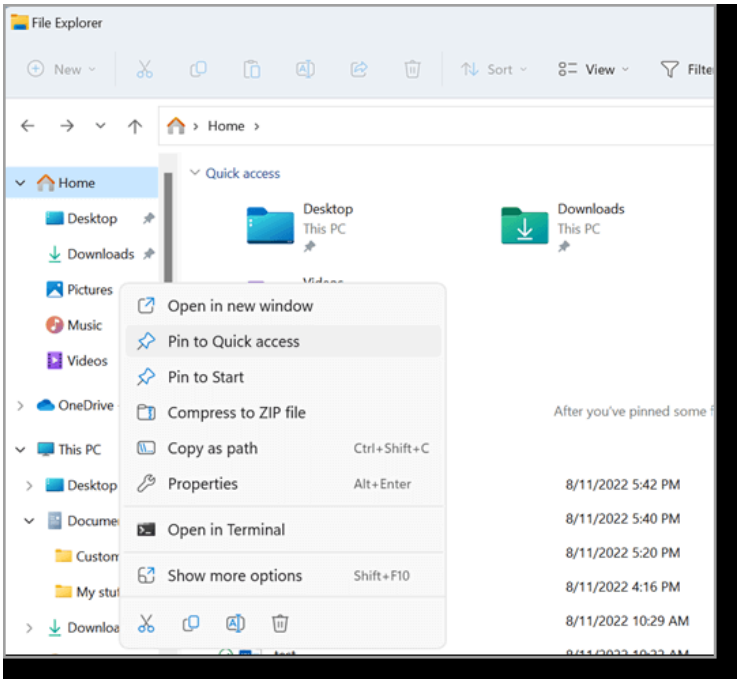
[215] https://winbuzzer.com/2020/03/07/windows-10-how-to-get-the-windows-experience-index-wei-score-xcxwbt/
[216] https://webmarks.info/howto/
[217] https://learn.microsoft.com/en-us/windows/win32/winsat/win32-winsat
[218] https://techtipsinfinite.blogspot.com/2020/05/blog-post.html

File Explorer (previously Windows Explorer)

By using “File Explorer” users can access/manipulate files very quickly - as shown in the screenshot below. We can open the “File Explorer” using the shortcut “WinKey+E”[219].

Also, since Windows 10 “Windows Explorer” name was changed to “File Explorer”[220]. Moreover, it is part of the graphical shell which is started when a user logs on to the system - “explorer.exe”[221]. We can think about it like “/bin/bash” in Linux but in this case it’s a graphical shell.

Lastly, “Windows Explorer” can be extended (Windows Shell Extensions), this is based on COM[222] objects. The shell extensions can be: toolbars, shell extensions handlers or a namespace extensions which allow certain folders to be displayed differently - check out “%windir%\Fonts” as an example[223].



[219] https://support.microsoft.com/en-us/windows/find-and-open-file-explorer-ef370130-1cca-9dc5-e0df-2f7416fe1cb1
[220] https://support.microsoft.com/en-us/windows/windows-explorer-has-a-new-name-c95f0e92-b1aa-76da-b994-36a7c7c413d7
[221] https://medium.com/@boutnaru/the-windows-process-journey-explorer-exe-windows-explorer-9a96bc79e183
[222] https://medium.com/@boutnaru/windows-com-component-object-model-71a76a97435c
[223] https://en.wikipedia.org/wiki/File_Explorer

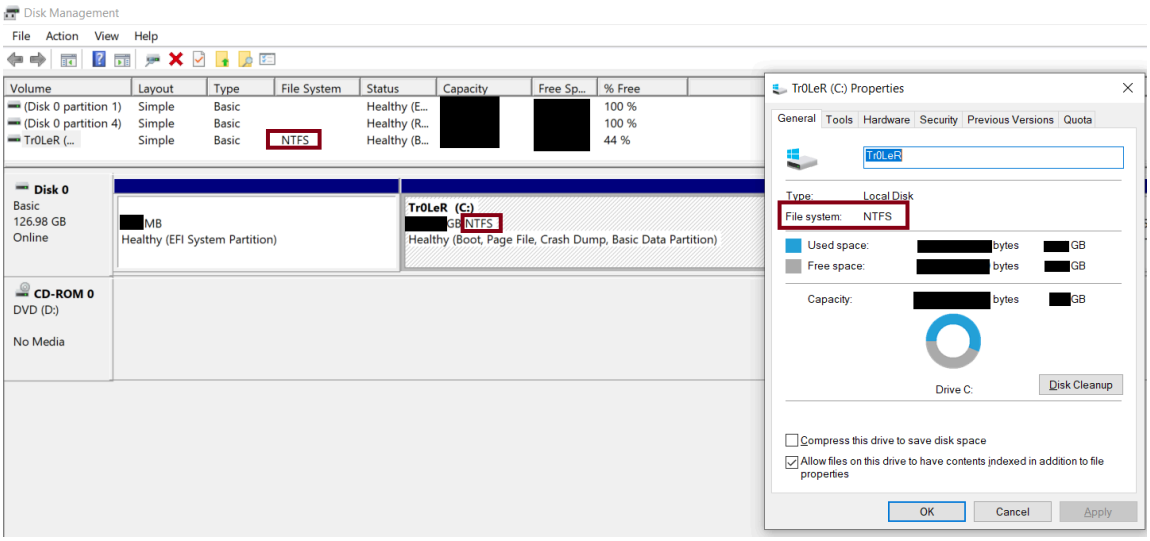
NTFS (New Technology File System)

NTFS (New Technology File System) is the primary file system used by Windows Workstation/Server versions in the last 30 years - as shown in the screenshot below. It was introduced as part of Windows NT 3.1 (1993) as a proprietary journaling file system by Microsoft[224].

Overall, it provides different features: increased reliability, increased security, support for large volumes, support for long file names and extended path names and flexible allocation of capacity, journaling, compression, hardlinks, volume shadow copy, transactions and quotas. I am going to detail only part of those features, for more information you can read the references[225].

Increased security is implemented by NTFS support file/folder permissions in using ACLs[226]: both DACLs[227] and SACLs[228]. Also, there is support for file encryption using EFS (Encrypting File System) or a part of disk encryption using Bitlocker.

Increased reliability is implemented by NTFS using log file and checkpoint information to restore consistency. Since Windows Server 2008 there is also a feature called "Self-Healing NTFS", which attempts to correct corruptions of the file system online. This is done without the need for "chkdsk.exe"[229].Lastly, we can also check out the reference implementation for NTFS from ReactOS[230].



[224] https://en.wikipedia.org/wiki/NTFS
[225] https://learn.microsoft.com/en-us/windows-server/storage/file-server/ntfs-overview
[226] https://medium.com/@boutnaru/the-windows-security-journey-acl-access-control-list-b7d9a6fe4282
[227] https://medium.com/@boutnaru/the-windows-security-journey-dacl-discretionary-access-control-list-c74545e472ec
[228] https://medium.com/@boutnaru/the-windows-security-journey-sacl-system-access-control-list-32488dcc80d7
[229] https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-r2-and-2008/cc771388(v=ws.10)
[230] https://github.com/reactos/reactos/tree/master/drivers/filesystems/ntfs

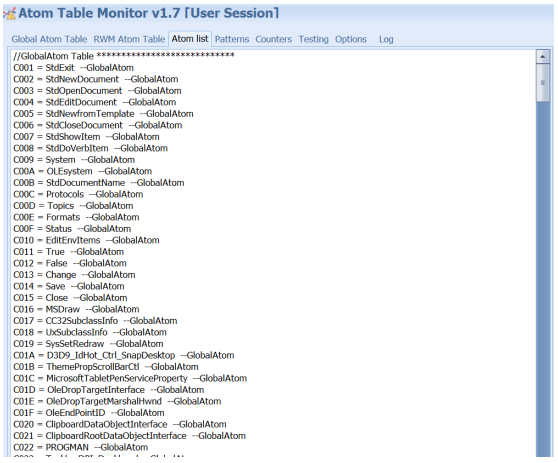
Atom Table

An “Atom Table” is a system-defined table which is used to store a string that has a specific identifier. When a program stores a string in an atom table it gets a 16 bit integer number (“atom”) which can be used to access the string (“atom name”). There is not one atom table[231].

Overall, there are two types of atom tables: “Global Atom Table” and “Local Atom Table”. The global table is defined for all processes on the system while the local one is private for the process which created it. By the way, there are two types of atoms: “String/Integer Atoms”[232].

In regards to “String Atoms”, they have specific properties. They can’t be longer than 255 bytes which can be returned within the range “0xC000-0xFFFF”. Also, the reference count is incremented when the string is added and decremented when removed. “Integer Atoms” are in the range of “0x0001-0xBFFF”, they don’t have any reference count or storage limitation[233].

Moreover, we can use different Win32 API calls to manipulate an “Atom String”. In regards of the global table we can use the following functions to add and find them: “GlobalAddAtomA” /”GlobalAddAtomW”[234] and “GlobalFindAtomA”/”GlobalFindAtomW”[235]. In regards to the local table we can use the following functions: “AddAtomA”/”AddAtomW”[236] and “FindAtomA”/”FindAtomW”[237]. Lastly, we can use the “atom-table-monitor”[238] for viewing the global atom table - as shown in the screenshot below. Also, we can go over a reference implementation of “Atom Management”[239] and “Executive Atom Functions”[240] as part of ReactOS.



[231] https://learn.microsoft.com/en-us/windows/win32/dataxchg/about-atom-tables
[232] https://www.oreilly.com/library/view/microsoft-windows-2000/0672319330/0672319330_ch23lev1sec1.html
[233] https://bsodtutorials.wordpress.com/2015/11/11/understanding-atom-tables/
[234] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-globaladdatomw
[235] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-globalfindatomw
[236] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-addatomw
[237] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-findatomw
[238] https://github.com/JordiCorbilla/atom-table-monitor/
[239] https://github.com/reactos/reactos/blob/master/sdk/lib/rtl/atom.c
[240] https://github.com/reactos/reactos/blob/master/ntoskrnl/ex/atom.c

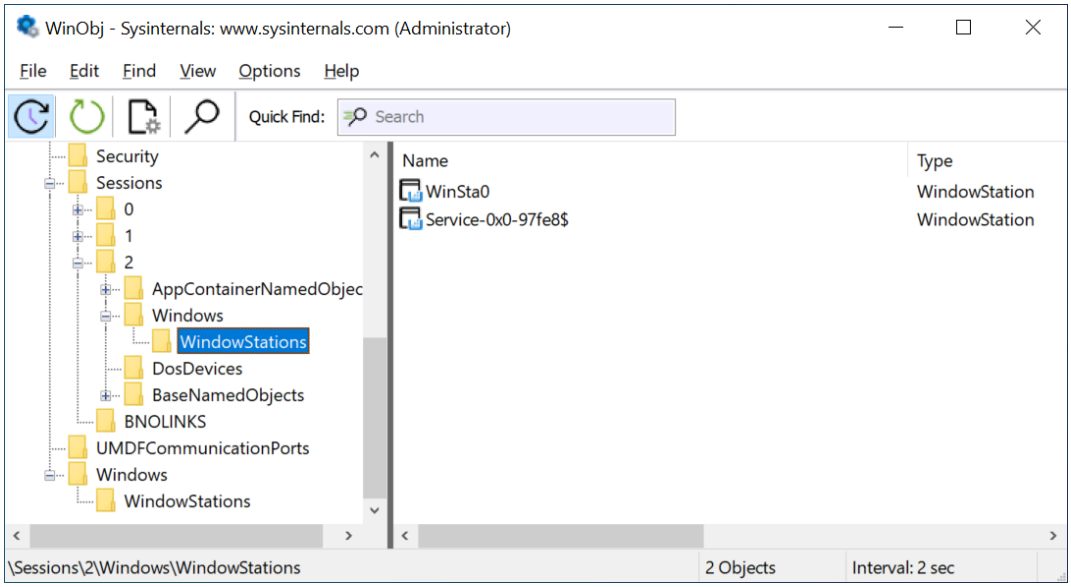
Window Station

"Window Station" contains: clipboard[241] , atom table and one or more desktop objects. Also, each window station is a securable object[242] which at creation is associated with the current session[243] of the calling process.

Moreover, the "interactive window station" is the only window station that can receive input or display a UI (user interface). This window station is named "WinSta0" - as shown in the screenshot from Sysinternals' WinObj below. Thus, when an interactive logon is performed the display device, mouse and the keyboard are assigned to the window station[244].

In case of a remote connection (using "Remote Desktop Services"), a new session is started for each user that logs on. Every session contains a collection of "Windows Station", a clipboard and more[245]. Because of that, each session that is created for a remote connection gets its own interactive window station named "WinSta0".

Lastly, we can use the Win32 API call "CreateWindowStationA"[246] or "CreateWindowStationW"[247]. We can also go over a reference implementation of those functions as part of ReactOS[248]. Also, you can over the kernel part "NtUserCreateWindowStation"[249].



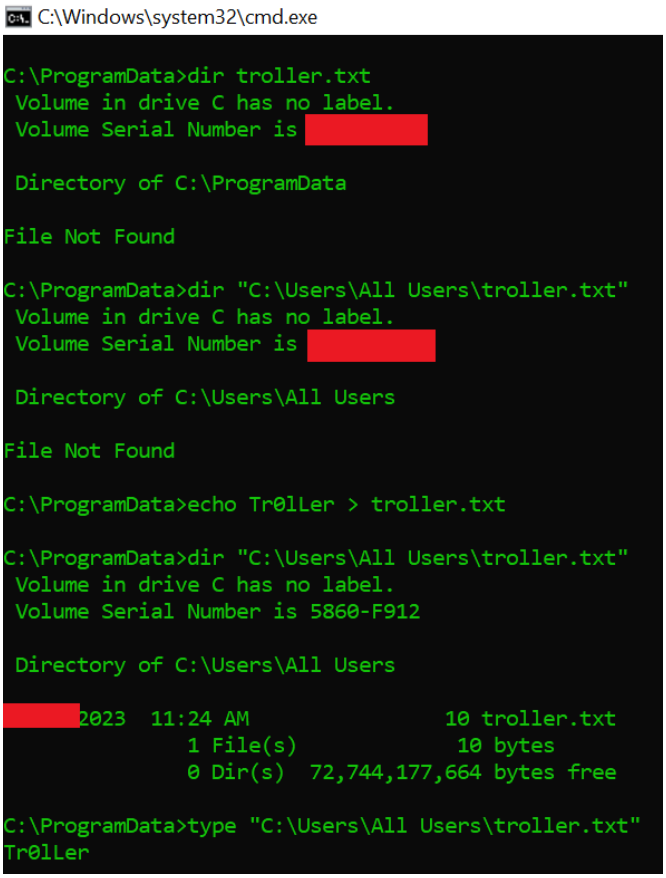
[241] https://medium.com/@boutnaru/windows-clipboard-c63e369d35d9
[242] https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad
[243] https://medium.com/@boutnaru/windows-sessions-fd5330911d54
[244] https://learn.microsoft.com/en-us/windows/win32/winstation/window-stations
[245] https://brianbondy.com/blog/100/understanding-windows-at-a-deeper-level-sessions-window-stations-and-desktops
[246] https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-createwindowstationa
[247] https://learn.microsoft.com/en-us/windows/win32/api/winuser/nf-winuser-createwindowstationw
[248] https://github.com/reactos/reactos/blob/master/win32ss/user/user32/misc/winsta.c
[249] https://github.com/reactos/reactos/blob/master/win32ss/user/ntuser/winsta.c#L731

ProgramData Directory

"ProgramData" is a hidden directory (normally "C:\ProgramData") which holds data that is not relevant for a specific user only. As opposed to the "Program Files" directory, it can be used by applications to store data for standard users, but does not require elevated permissions[250]. There is also an environment variable pointing to this folder ("echo %ProgramData%").

Thus, it contains data that is shared between users. Of course, using DACLs the access to folders/files can be limited. On Windows XP there wasn't any "ProgramData" directory, instead there was "C:\Documents and Settings\All Users\Application Data"[251]. Due to that, if we access "C:\Users\All Users" we are redirected to "C:\ProgramData" - as shown in the screenshot below.

Lastly, the owner of the directory is SYSTEM. By default, every user/group that is part of the local administrators group has full control on the folder, while standard users care to read & execute[252].



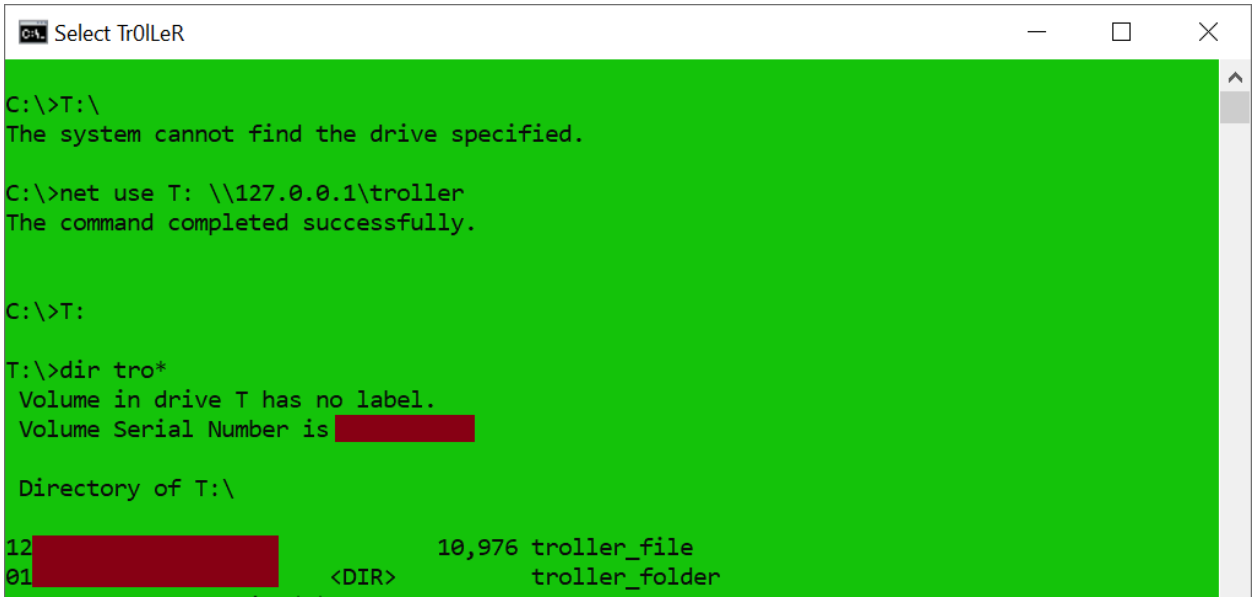
[250] https://learn.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/microsoft-windows-shell-setup-folderlocations-programdata
[251] https://www.howtogeek.com/278562/what-is-the-programdata-folder-in-windows/
[252] https://www.kapilarya.com/what-is-programdata-folder-in-windows

Windows Shares

In general a shared folder on Windows allows users to access resources over the network. In order for that to work both the client (accessing the share) and the server (the entity which holds the share) must have the file and print services for Windows enabled. It is important to understand that we can't share a specific file, thus if we want to share a file we need to share the entire directory which contains it[253].

Moreover, shared folders are accessed using UNC (Universal Naming Convention) paths. The pattern is as follows: "\\[SERVER]\[SHARE_NAME]". The "[SERVER]" part can be an FQDN (Fully Qualified Domain Name), NetBIOS name, IPv4 address or an IPv6 address. By the way, when accessing a share from a Linux/macOS client we use the "smb://" prefix[254].

Lastly, there are different ways to share a directory using the "File Explorer" ("explorer.exe") without or with the need of using passwords - aka password protected sharing[255]. At the end a shared folder is a folder hosted on a different Windows machine which we can access remotely. We can also assign a shared folder to a specific drive letter using "net.exe"[256] - as shown in the screenshot below.

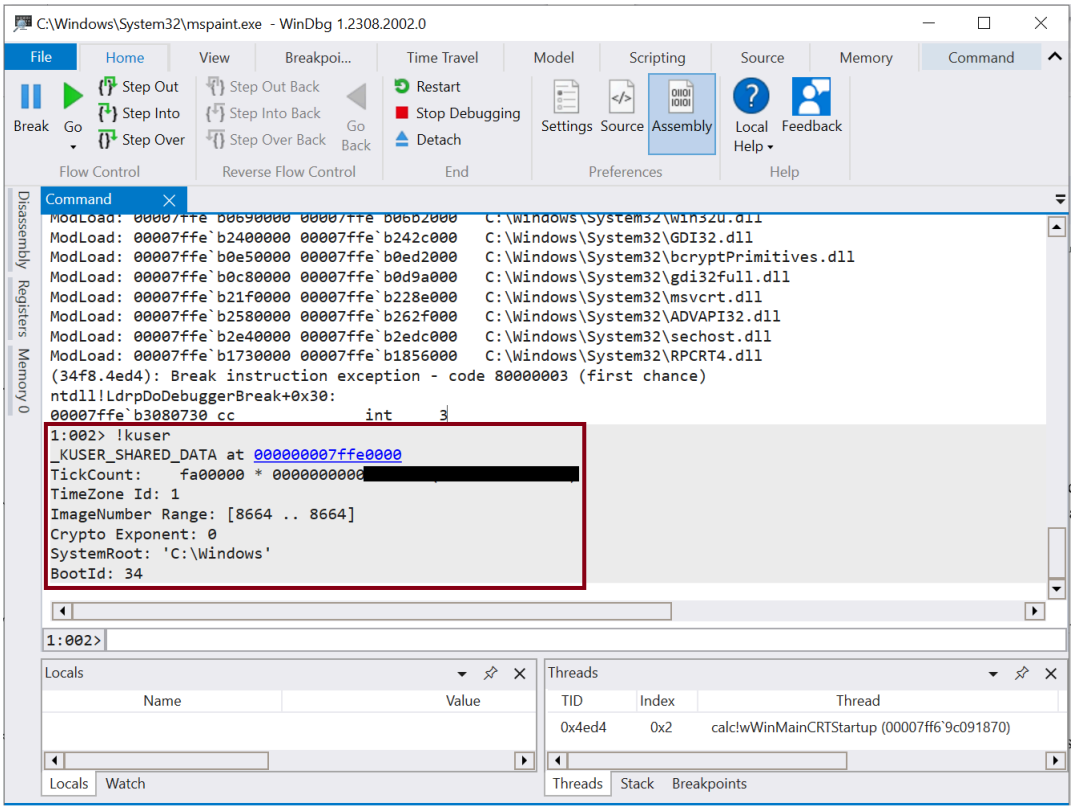


[253] https://www.free-online-training-courses.com/shared-folders-unc-paths/
[254] https://it.cornell.edu/shared-file/connect-windows-file-share-unc-path
[255] https://pureinfotech.com/setup-network-file-sharing-windows-10/
[256] https://medium.com/@boutnaru/the-windows-process-journey-net-exe-net-command-91e4964f20b8

SharedUserData (KUSER_SHARED_DATA)

KUSER_SHARED_DATA is a single page which is mapped in a fixed address both in user-mode and kernel mode. It is mapped into the address space of every process in order to provide a quick way to obtain global data (interrupt time, processor extensions, debugger state, version, etc). The user-mode mapping has read-only permissions while the kernel one has read+write permissions[257]. By the way, it resembles vsyscall/vDSO in Linux in that case.[258]

Moreover, the user-mode read-only address for the shared user data in both 64-bit or 32-bit is "0x7FFE0000". We can also see that in the source code or ReactOS[259] or in the screenshot shown below. In the case of the read+write kernel address it is "0xFFDF0000" for 32-bit Windows or "0xFFFFF780`00000000" for 64-bit Windows[260] - for security reasons there changes were made in the kernel definition. Lastly, we can use the "!kuser" extension of WinDBG in order to show the shared user-mode page[261]- as shown in the screenshot below.



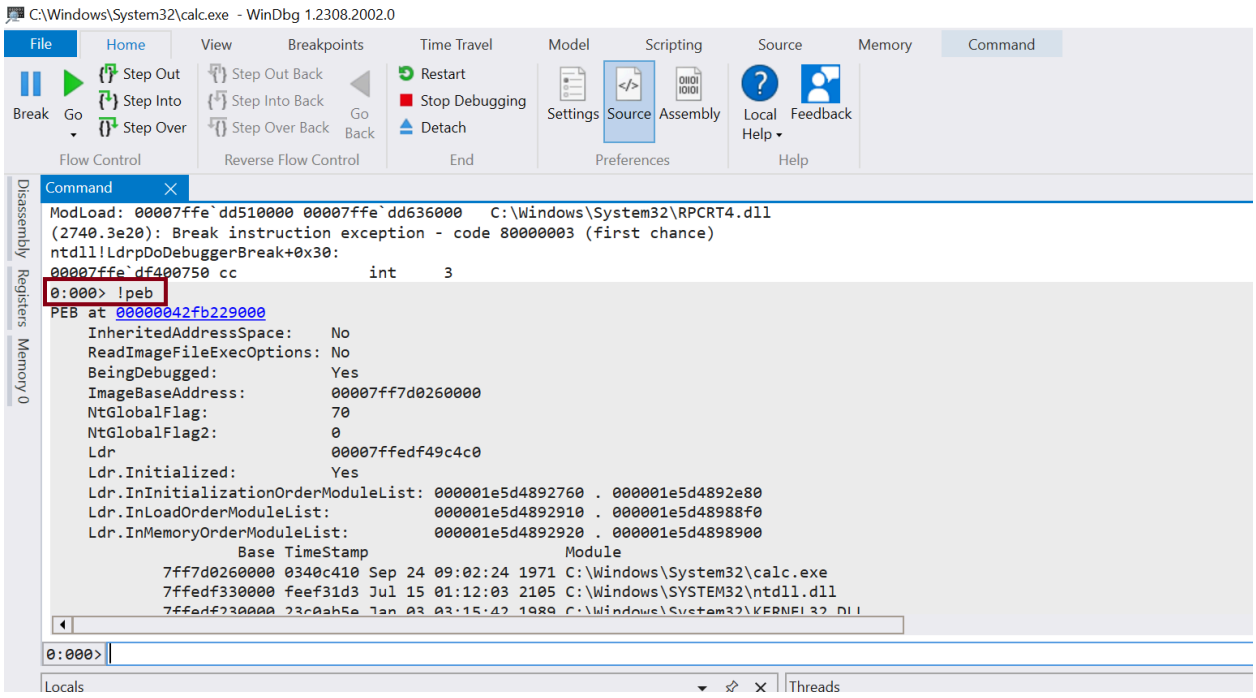
[257] https://msrc.microsoft.com/blog/2022/04/randomizing-the-kuser_shared_data-structure-on-windows/
[258] https://man7.org/linux/man-pages/man7/vdso.7.html
[259] https://github.com/reactos/reactos/blob/master/modules/rostests/winetests/ntdll/time.c#L237
[260] https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/api/ntexapi_x/kuser_shared_data/index.htm
[261] https://learn.microsoft.com/en-us/windows-hardware/drivers/debuggercmds/-kuser

PEB (Process Environment Block)

PEB (Process Environment Block) has been part of the Windows operating system since Windows 2000. It contains information about a "running" process like: the process name, PID and loaded modules. The loaded modules are every PE (Portable Executable) file loaded into the memory address space of the processes - which are the main binary and all the DLLs[262].

Moreover, we can access the PEB structure from user-mode, as opposed to EPROCESS that we can access from kernel-mode only (unless there is some security bug). We can see the information using a debugger like WinDbg (using the "!peb" command) - as shown in the screenshot below. By the way, every process has its own PEB[263].

Lastly, I wanted to point out some interesting fields which are part of the PEB. "IsLongPathAwareProcess", which states if the process is aware of paths with more than 260 characters. "ImageBaseAddress", which states the virtual memory address at which the executable was loaded into memory. "Object Table" which is the handle table of the process[264]. More regarding the other fields in future writeups.



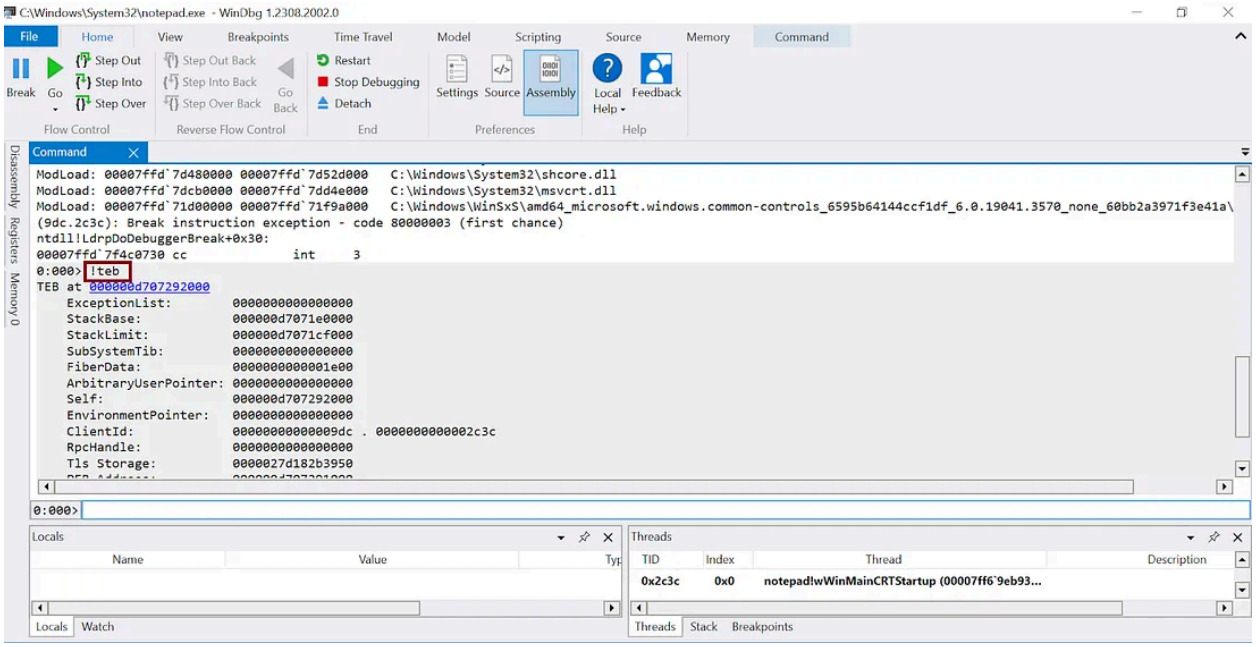
[262] https://mohamed-fakroud.gitbook.io/red-teamings-dojo/windows-internals/peb
[263] https://ntopcode.wordpress.com/2018/02/26/anatomy-of-the-process-environment-block-peb-windows-internals/
[264] https://github.com/Faran-17/Windows-Internals/blob/main/Processes%20and%20Jobs/Processes/PEB%20-%20Part%201.md

TEB (Thread Environment Block)

As Windows has the data structure PEB[265] which is responsible for holding information about processes[266], there is also TEB (Thread Environment Block) which holds relevant information about threads[267].

Overall, we can access the TEB structure from user-mode, as opposed to ETHREAD that we can access from kernel-mode only (unless there is some security bug). So we can state that it holds the thread's user-mode representation, which is not used by the kernel[268].

Moreover, there are two data structures for TEB one for x64/64 bit (_TEB64) and for x86/32 bit (_TEB32), the same is also relevant for PEB — more about this when talking about WOW64. As opposed to EPROCESS/ETHREAD which are linked between each other among the entire system TEB/PEB are not. Thus, we can't travers all the processes/threads by using them[269].Lastly, we can see the information using a debugger like WinDbg (using the "!teb" command) - as shown in the screenshot below.



[265] https://medium.com/@boutnaru/the-windows-concept-journey-peb-process-environment-block-e8078a146612
[266] https://medium.com/@boutnaru/windows-process-923b9332c12
[267] https://medium.com/@boutnaru/windows-threads-3a839fa67ae3
[268] https://www.geoffchappell.com/studies/windows/km/ntoskrnl/inc/api/pebteb/teb/index.htm
[269] https://stackoverflow.com/questions/74260342/thread-environment-block-and-process-environment-block

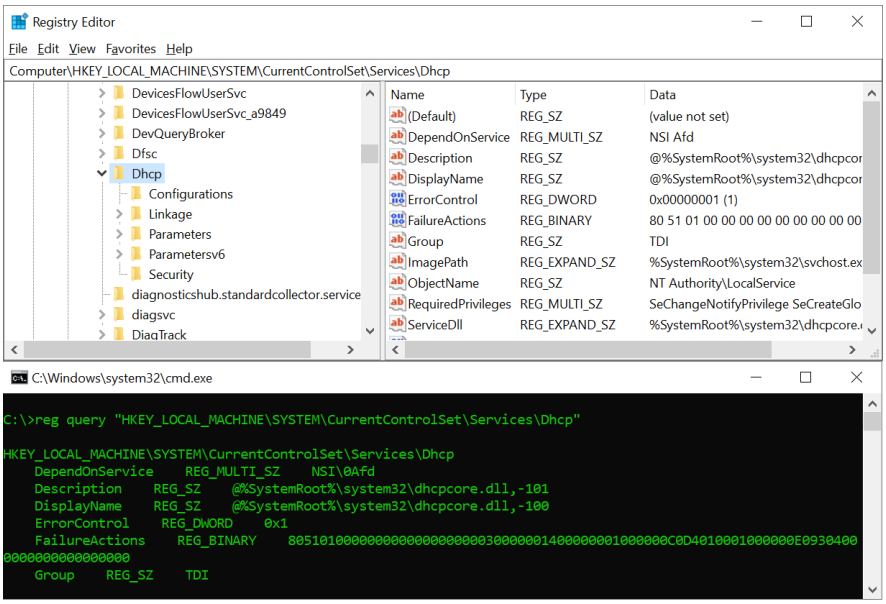
Registry

We can say that the "Registry" is a central hierarchical database used by Windows to store configuration of the operating system/hardware/devices/application/etc (and other types of information). It is referenced by Windows constantly during normal operation for various tasks. The registry replaced most of the text-based "*.ini" files that were in used in Windows 3.x and other files like "config.sys" and "autoexec.bat" in MS-DOS[270].

Overall, the registry is composed of hives, that are groups of keys that contain different values. There are 6 different hives (aka root keys): HKEY_CLASSES_ROOT, HKEY_CURRENT_USER, HKEY_LOCAL_MACHINE, HKEY_USERS, HKEY_CURRENT_CONFIG and HKEY_PRERFORMANCE_DATA[271].By the way, part of them are virtual hives - more on that and details about each one of them are part of future writeups.

Moreover, there is a limit of 64K in size of values of a key. Each value in a key has one of the following types: binary value (REG_BINARY), dword value (REG_DWORD), expandable string value (REG_EXPAND_SZ), multi-string value (REG_MULTI_SZ), symbolic link (REG_LINK), null terminated string (REG_SZ) and more[272].

Lastly, we can access the registry (for reading/writing data) using different interfaces/tools like: the Win32 Registry API[273], "reg.exe" and "regedit.exe" - as shown in the screenshot below.



[270] https://learn.microsoft.com/en-us/troubleshoot/windows-server/performance/windows-registry-advanced-users
[271] https://en.wikipedia.org/wiki/Windows_Registry
[272] https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry-value-types
[273] https://learn.microsoft.com/en-us/windows/win32/sysinfo/registry-functions

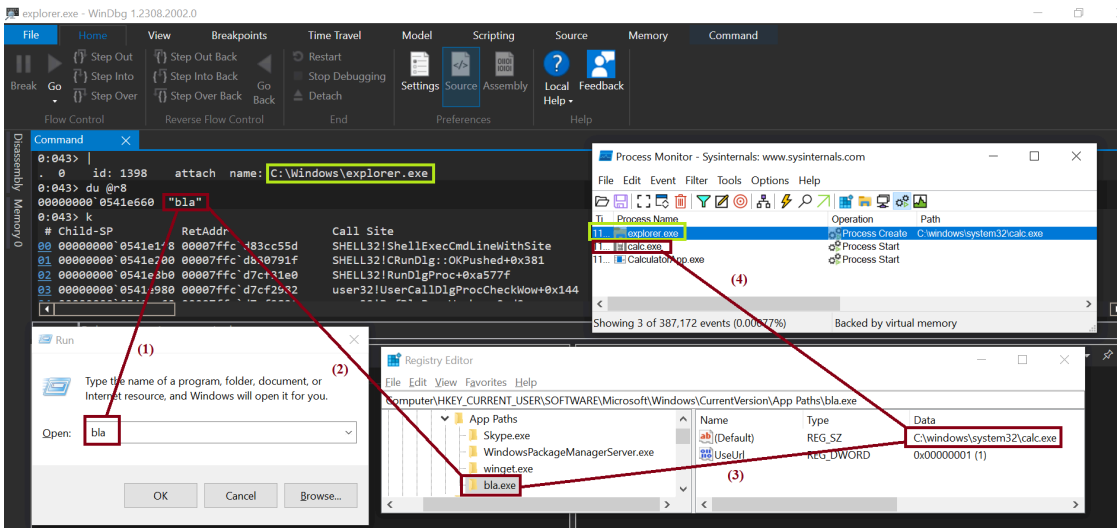
App Paths (Application Registration)

Application Registration (aka “App Paths”) is a registry[274] key used Windows in order to provide a private search path for specific “*.exe”/”*.dll” files. The location of the key is the following: “HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\”[275]. We also have a counterpart in “HKCU\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\”[276].

Thus, we can say that “App Paths” has two major goals. First, to map an executable name (like “App.exe”) to the program's fully qualified path. Second, appending information to the PATH environment variable. It is important to understand that executable registered as a subkey can be launched using the “start” command in “cmd.exe” even if they are not found using PATH [277].

Overall, “App Paths” is checked as part of the flow of “ShellExecute”\”ShellExecuteEx”[278] that is for finding the current application - as shown in the screenshot below. We can review that in the reference implementation as part of ReacOS[279].

Lastly, when creating a sub key as part of the application registration we can use one of 6 entries: “(Default)” (the full path of the application), “DontUseDesktopChangeRouter” (which have to be used in a case of a debugger to avoid deadlocks), “DropTarget” (the CLSID of an object that implements IDropTarget), “Path” (string to append to the PATH environment variable), “SupportedProtocols” (URL protocol schemes for a given key) and “UseUrl” (indicates that your application can accept a URL)[280].



[274] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[275] https://docs.revenera.com/installshield28helplib/helplibrary/PA_AppPaths.htm
[276] https://helgeklein.com/blog/how-the-app-paths-registry-key-makes-windows-both-faster-and-safer/
[277] https://renenyffenegger.ch/notes/Windows/registry/tree/HKEY_LOCAL_MACHINE/Software/Microsoft/Windows/CurrentVersion/App-Paths/index
[278] https://learn.microsoft.com/en-us/windows/win32/shell/launch
[279] https://github.com/reactos/reactos/blob/master/dll/win32/shell32/shlexec.cpp#L762
[280] https://learn.microsoft.com/en-us/windows/win32/shell/app-registration

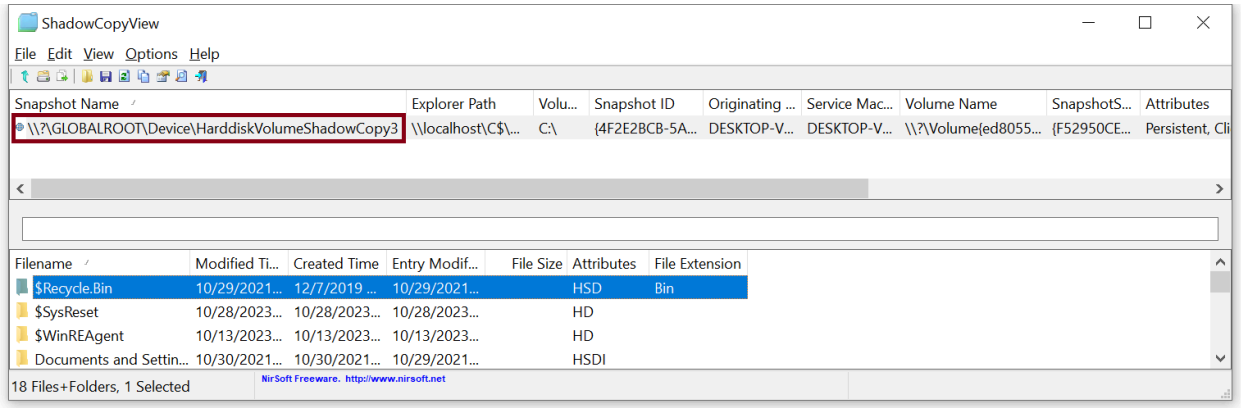
Shadow Copy

"Shadow Copy" is also known as "Volume Shadow Copy"/"Volume Shadow Copy Service" (VSS). It is a technology included as part of the Windows operating system. By using it, users/the OS can create backup copies/snapshots of files/volumes even if they are in use by an application/the OS. "Shadow Copy" requires an NTFS/ReFS file system. The shadow copies can be created locally or external (removable media or network) volumes[281].

Overall, "Shadow Copy" is used for managing running volumes and creating snapshots of them when requested. The provider is a software component that receives the request for creating the shadow copy and then signals about the upcoming copy, creates it and maintains it until it is not needed anymore. There are three types of provider: system (which is the default one), hardware (which works together with an hardware device like a RAID controller) and software that is implemented as a DLL/kernel device[282].

Moreover, each shadow copy is identified by a GUID (Global Unique Identifier). There is also a shadow copy set, which is a collection of shadow copies of various volumes all taken at the same time - it is also identified by a GUID[283]. When creating a shadow copy a new device is created in the pattern of HarddiskVolumeShadowCopy[NUM], where NUM is the index of the snapshot - you can check it out using Sysinternals' WinObj tool[284].

Lastly, examples of components which are based on "Shadow Copy" are: "Backup and Restore" (which was deprecated since Windows 8) and "System Restore". We can use the "ShadowCopyView" tool by NirSoft[285] to view the created snapshots for Windows Vista/7/8/10/etc. Also, we can use it to browse the older versions of the files, copy them into a folder on a disk and checkout metadata information about the shadow copy itself - as also shown in the screenshot below.



[281] https://en.wikipedia.org/wiki/Shadow_Copy
[282] https://learn.microsoft.com/en-us/windows/win32/vss/providers
[283] https://learn.microsoft.com/en-us/windows/win32/vss/shadow-copies-and-shadow-copy-sets
[284] https://learn.microsoft.com/en-us/sysinternals/downloads/winobj
[285] https://www.nirsoft.net/utils/shadow_copy_view.html

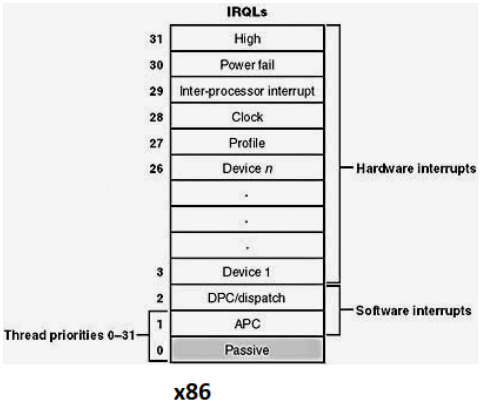
IRQL (Interrupt Request Level)

An IRQL (Interrupt Request Level) basically defines the hardware priority that the CPU operates on at a specific time. As part of the "Windows Driver Model" a thread running at a low IRQL can be interrupted by code at a higher IRQL[286].

Overall, on x86 systems the IRQL ranges is 0-31, while on x64 the IRQL range is 0-15 - as shown in the tables below[287]. For example "IRQL 0" means the CPU is running normal user mode/kernel code, it is aka "PASSIVE_LEVEL". "IRQL 1" means the CPU is running an APC (Asynchronous Procedure Call)/page fault, it is aka "APC_LEVEL". "IRQL 2" is used for DPC (Deferred Procedure Call)/thread scheduling, it is aka "DISPATH_LEVEL"[288].

Moreover, in case of a multi-processor system each CPU can execute code at a different IRQL. Thus, using just IRQL for synchronization could be problematic[289]. By the way, we can use WinDBG (while debugging the kernel) to display the IRQL of a specific processor using the "!irql" extension[290].

Lastly, IRQL is something that every driver/kernel developer in Windows should consider, however it is not something that user-mode developers should think about.



x86

	x64	IA64
15	High/Profile	High/Profile/Power
14	Inter-processor interrupt/Power	Inter-processor interrupt
13	Clock	Clock
12	Synch	Synch
11	Device *n*	Device *n*
		⋮
4	⋮	Device 1
3	Device 1	Correctable Machine Check
2	Dispatch/DPC	Dispatch/DPC & Synch
1	APC	APC
0	Passive/Low	Passive/Low

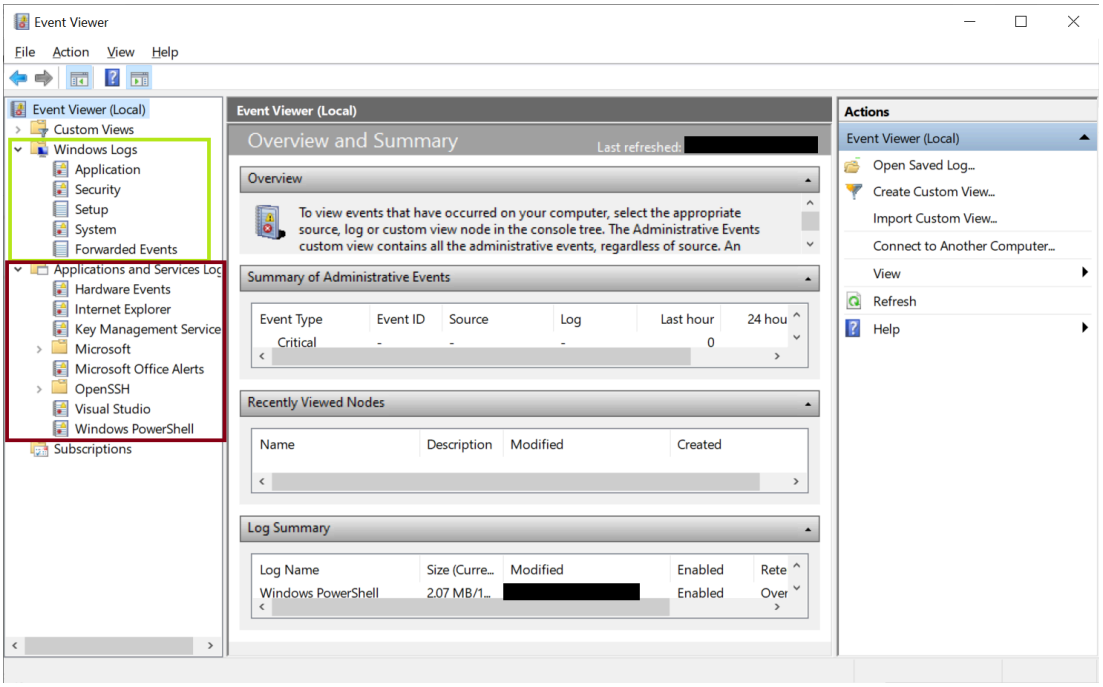
[286] https://techcommunity.microsoft.com/t5/ask-the-performance-team/what-is-irql-and-why-is-it-important/ba-p/372666
[287] https://bsodtutorials.blogspot.com/2013/10/interrupt-dispatch-table-idt.html
[288] https://learn.microsoft.com/en-us/windows-hardware/drivers/kernel/managing-hardware-priorities
[289] https://download.microsoft.com/download/e/b/a/eba1050f-a31d-436b-9281-92cdfeae4b45/IRQL_thread.doc
[290] https://learn.microsoft.com/en-us/windows-hardware/drivers/debuggercmds/-irql

Windows Event Logs

Windows event log is a record of events which are stored by the operating system. We can classify the Windows event logs to 5 categories: “Application Logs” (event logged by applications), “System Logs” (hardware/driver related events), “Setup Logs” (events that have occurred during installations), “Security Logs” (security related events like failed/successful logons and logoffs) and “Forward Events” that defines which events are going to be logged in other computers in the network[291].

Moreover, we can use the “Event Viewer” (“%windir%\System32\eventvwr.msc”) in order to check out the different Windows event logs. As we can see the 5 categories of logs described above are not the only logs supported in Windows. They appear under the “Windows Logs” section while the others appear under “Applications and Services Logs” (where we can see logs for Powershell, hardware events and more) - both are marked in the screenshot below.

Lastly, the event log files have an “.evtx” extension (until Windows 7 they had “.evt” extension). By default they are stored in the “%windir%\System32\winevt\Logs” directory. Unlike traditional Unix logs which are text based (excluding journalctl) the Windows event logs store events in binary format[292].



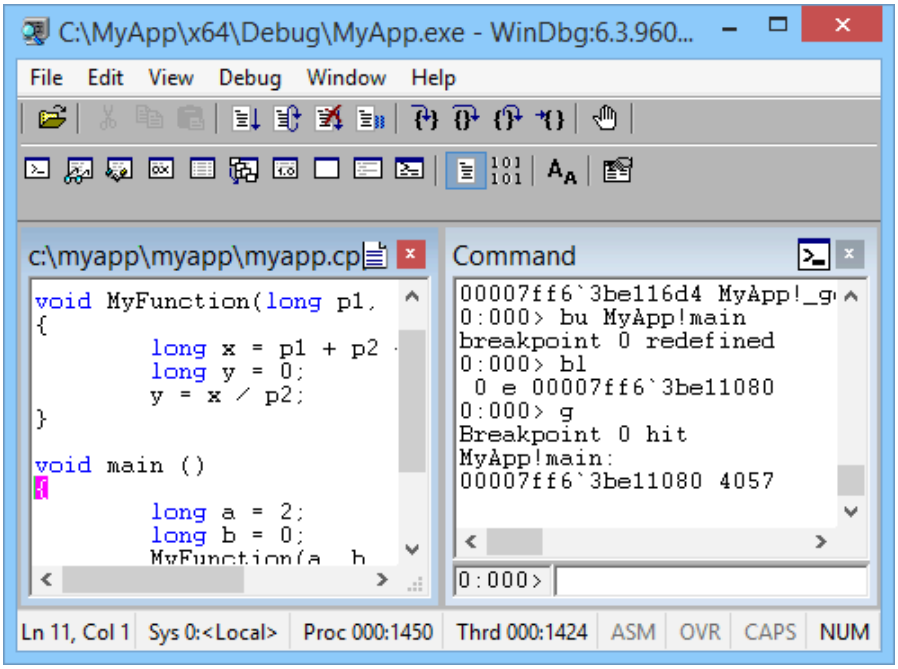
[291] https://www.manageengine.com/products/active-directory-audit/learn/windows-event-log.html
[292] https://eventlogxp.com/essentials/windowseventlog.html

PDB (Program Database) Files

In general, PDB (Program Database) is a file format developed by Microsoft which is used for storing debug information regarding applications. Those files (usually with "*.pdb" extension) are created during the compilation process. The information included might contain the following: symbols, line number of the symbol, original file name and more[293]. Thus, when using a debugger it can leverage the data as part of the PDB file to locate symbols and correlate them to the execution state - as shown in the screenshot below taken using WinDbg[294].

Overall, back with 16-bit versions of Visual C++ the debugging information was stored at the end of the ".exe" (Executable) ".dll" (Dynamic Link Library) file by the linker. As mentioned above today the information is stored in a separate PDB file and the binary contains the name/full path[295].

Moreover, Microsoft's "Debug Interface Access Software Development Kit" (DIA SDK) provides access to the debug information stored in PDB files[296]. We can also go over a reference implementation from RactOS which parses information from PDB files[297]. Lastly, I recommend going over the Github repository "microsoft-pdb" which contains information from Microsoft about the PDB format[298].



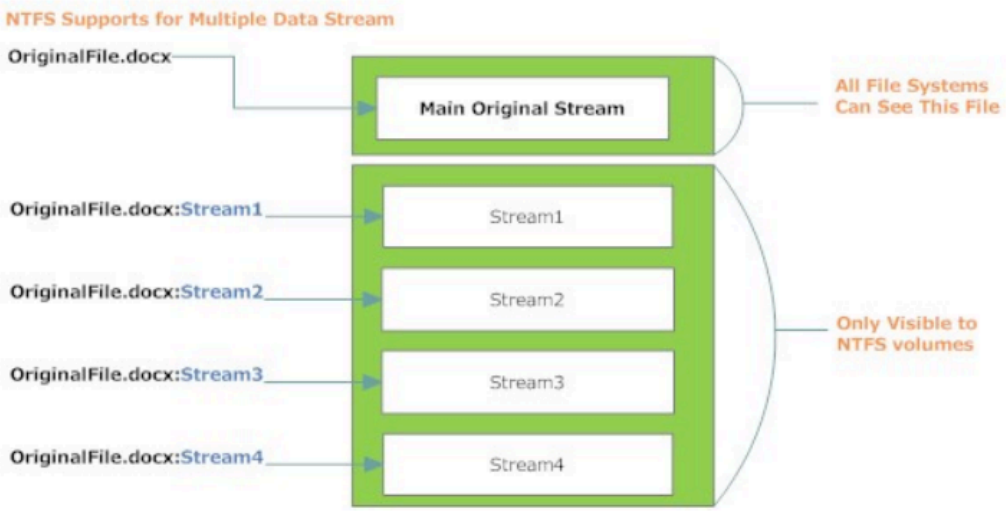
[293] https://en.wikipedia.org/wiki/Program_database
[294] https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/getting-started-with-windbg
[295] https://web.archive.org/web/20150530212051/https://support.microsoft.com/en-us/kb/121366
[296] https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2015/debugger/debug-interface-access/debug-interface-access-sdk
[297] https://github.com/reactos/reactos/blob/master/dll/win32/dbghelp/msc.c#L3
[298] https://github.com/Microsoft/microsoft-pdb

ADS (Alternate Data Stream)

ADS (Alternate Data Stream) is a feature of NTFS[299] that has been included in Windows in order to provide compatibility with files stored on a Mac file system. By using ADS files that can contain more than one stream of data (there is at least one), the default one is called ":$DATA"[300].

Thus, by leveraging ADS Windows servers can act as file servers for Apple based computers. With the support for multiple streams a Mac user can copy files form/to a Windows server without losing any resource information. By the way, there are also archive/backup software who use ADS to store file revision history[301].

Lastly, by default we accessing a file the mainstream is used as opposed to the other streams -as shown in the diagram below[302]. Also, we can use the "\R" flag of "cmd.exe"[303] or "streams.exe"\"streams64.exe" from the Sysinternals Suite[304] in order to display alternate data streams of the file.



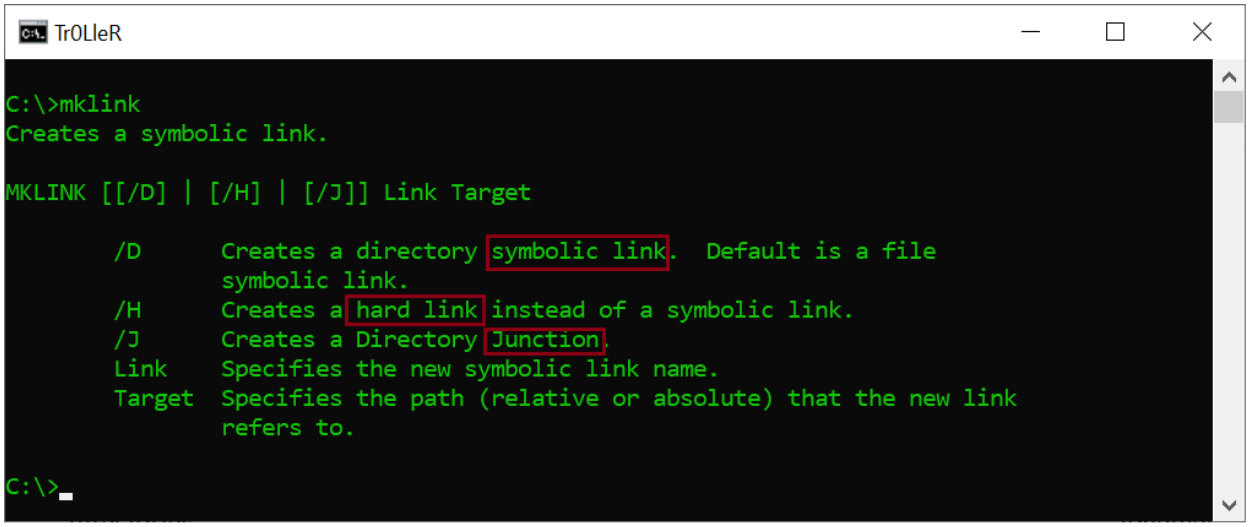
[299] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-new-technology-file-system-433e27a2256a
[300] https://owasp.org/www-community/attacks/Windows_alternate_data_stream
[301] https://blog.netwrix.com/2022/12/16/alternate_data_stream/
[302] https://web.archive.org/web/20230424001002/https://www.darknessgate.com/security-tutorials/date-hiding/ntfs-alternate-data-streams/
[303] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b
[304] https://learn.microsoft.com/en-us/sysinternals/downloads/streams

NTFS File Links

As part of the NTFS[305] file system Microsoft has implemented different types of linking capabilities (since Windows NT 4.0). This provides users a convenient method for users to access their data. Over the years those capabilities were improved and now can be used to link files and/or directories together[306].

Overall, a file link is a file which is used in order to point to a file/directory. The link is sometimes called as the source and the file/directory pointed to is called sometimes the target of the link. Links are supported not only on Windows (using NTFS) but also other operating systems (and their filesystems) like Linux, FreeBSD and macOS[307].

Lastly, we have three main links types supported by NTFS: symbolic links, hard links and junctions - more on each one of them in future writeups. We can create each one of them using the "mklink.exe" command line utility[308] - as shown in the screenshot below.



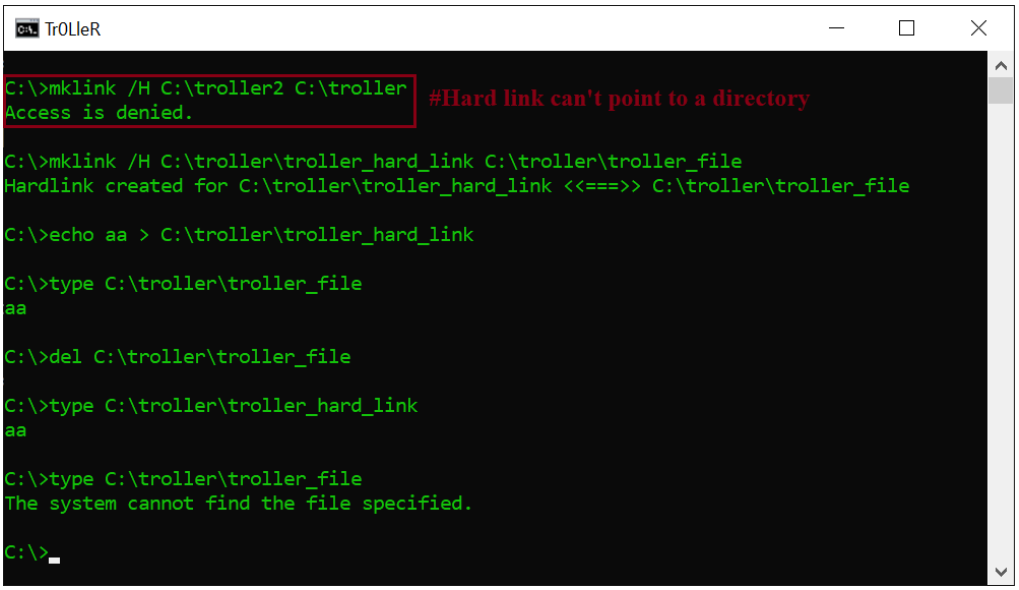
[305] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-new-technology-file-system-433e27a2256a
[306] https://www.2brightsparks.com/resources/articles/ntfs-hard-links-junctions-and-symbolic-links.html
[307] https://en.wikipedia.org/wiki/Symbolic_link
[308] https://ss64.com/nt/mklink.html

NTFS Hard Links

A hard link is one of the file link types[309] supported by NTFS[310]. The goal of a hard link is to represent another file on the same volume without the need of duplicating the data. We can create more than one hard link to point to the same file. It is important to know there is not support for hard links on directories to avoid inconsistencies in parent directory entries - as shown in the screenshot below[311].

Moreover, every change to the content/attributes of the file propagates to all hard links. However, the directory size/attribute information of the file is only visible only at the link through which the change was made. When deleting an file\hard link (DeleteFileA\DeleteFileW) the blocks containing the data of the file will only be deleted after all the hard links\files are removed[312] - as shown in the screenshot below.

Lastly, for creating an hard link we can use the CreateHardLinkA[313] or CreateHardLinkW[314] API call. It is important to understand that we can't point a hard link to a different partition/network drive. By the way, FAT and older versions of ReFS (until version 3.5) don't have support for hard links.



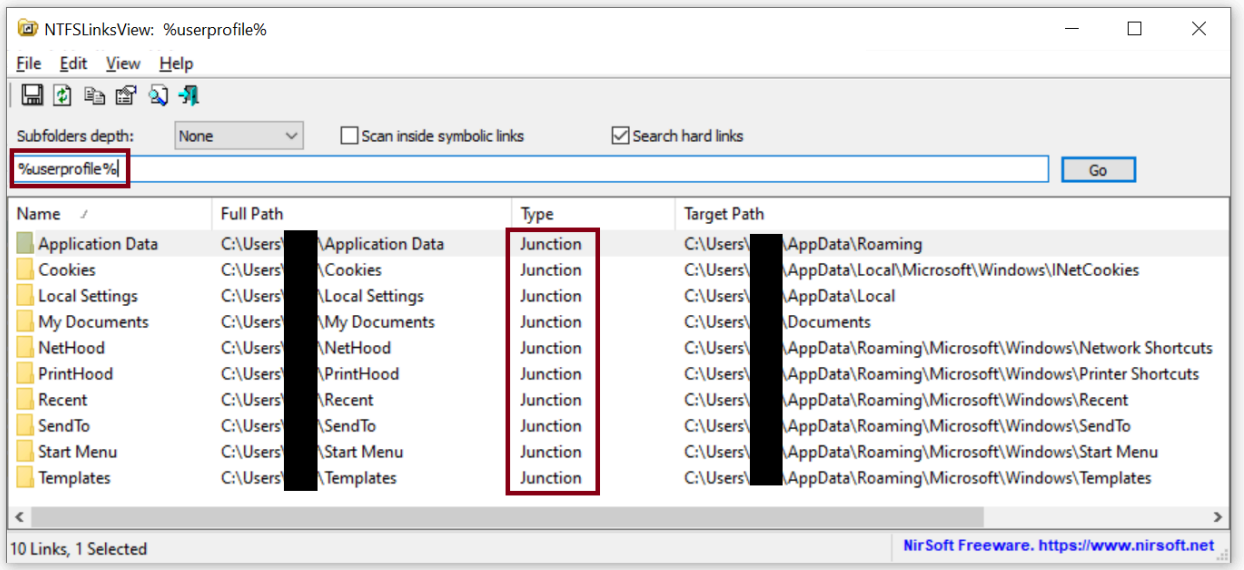
[309] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-file-links-55547033d8ae
[310] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-new-technology-file-system-433e27a2256a
[311] https://www.2brightsparks.com/resources/articles/ntfs-hard-links-junctions-and-symbolic-links.html
[312] https://learn.microsoft.com/en-us/windows/win32/fileio/hard-links-and-junctions
[313] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createhardlinka
[314] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createhardlinkw

NTFS Junctions

A junction is one of the file link types[315] supported by NTFS[316]. As opposed to "NTFS Hard Links"[317] a junction is a reference to a folder (hard link points to a file). Junctions can be link directories located on different partitions or volume (only locally on the same computer)[318].

Moreover, junctions are based on a NTFS feature called "reparse points" (introduced in NTFS 3.0 and shipped with Windows 2000). By the way, sometimes junctions are referred to as "soft links", which is different from a "Symbolic Link"[319]. We can say that junctions act as an alias for directories, which can be created using the "mklink.exe" utility[320].

Lastly, there is no need for admin privileges for creating a junction and it stores the absolute path of the target, thus issues with relative paths are not relevant[321]. Also, we can use "NTFSLinksView" from NirSoft for showing a list of all symbolic links and junctions in the specified folder, and their target paths[322]. Thus, we can use it to see examples for using junctions as part of the home folders of users (%userprofile%") - as shown in the screenshot below.



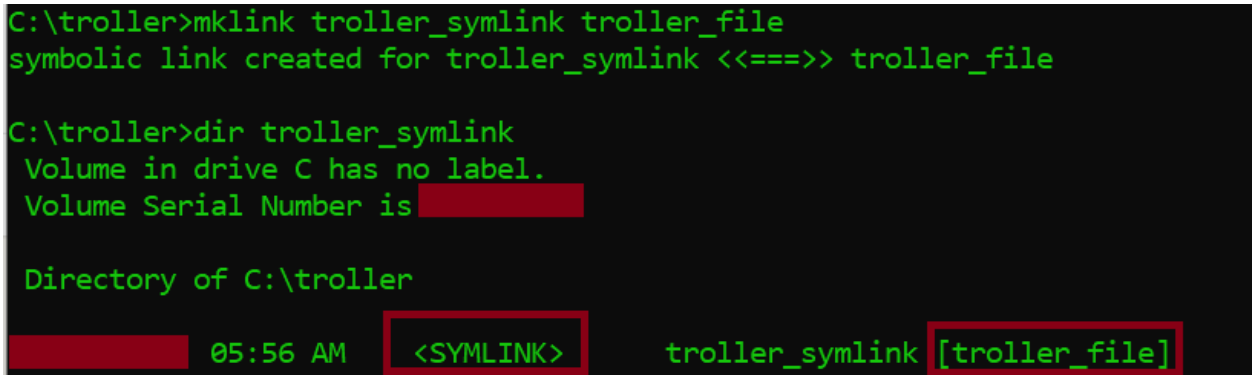
[315] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-file-links-55547033d8ae
[316] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-new-technology-file-system-433e27a2256a
[317] https://medium.com/@boutnaru/the-windows-concept-journey-8bc061672fe7
[318] https://www.2brightsparks.com/resources/articles/ntfs-hard-links-junctions-and-symbolic-links.html
[319] https://learn.microsoft.com/en-us/windows/win32/fileio/hard-links-and-junctions
[320] https://ss64.com/nt/mklink.html
[321] https://www.geeksforgeeks.org/ntfs-junction-points/
[322] https://www.nirsoft.net/utils/ntfs_links_view.html

NTFS Symbolic Links (symlinks)

A symbolic link (aka symlink) is one of the file link types[323] supported by NTFS[324]. As opposed to "NTFS Hard Links"[325] and "NTFS Junctions"[326] symbolic links can point both to files and folder which are local to the computer or remote[327]. Remote network locations are specified using SMB paths.

Overall, symbolic links were introduced in Windows Vista/Server 2008. A symbolic link can be pointed to a target based on an absolute path or a relative path (as opposed to hard links/junctions which support only absolute paths). In order to create a symbolic link we can use the API function "CreateSymbolicLinkA"/"CreateSymbolicLinkW" or the command line utility "mklink.exe"[328] - as shown in the screenshot below.

Lastly, since Windows 10 (insiders build 14972) symlinks can be created without the need for admin privileges. This is done by leveraging the "SYMBOLIC_LINK_FLAG_ALLOW_UNPRIVILEGED_CREATE" flag when using the API for creating a symlink. For "mklink.exe" to work without the need for admin privileges we should enable the "Developer Mode" feature[329].

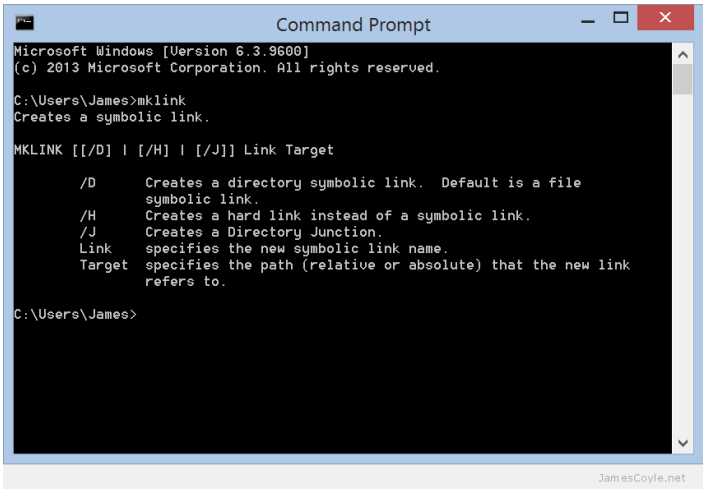


[323] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-file-links-55547033d8ae
[324] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-new-technology-file-system-433e27a2256a
[325] https://medium.com/@boutnaru/the-windows-concept-journey-8bc061672fe7
[326] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-junctions-bdbeb9aec611
[327] https://www.2brightsparks.com/resources/articles/ntfs-hard-links-junctions-and-symbolic-links.html
[328] https://learn.microsoft.com/en-us/windows/win32/fileio/creating-symbolic-links
[329] https://blogs.windows.com/windowsdeveloper/2016/12/02/symlinks-windows-10/

mklink (Make Link)

"mklink" is a builtin command of "cmd.exe"[330] which is used for creating different types of links. Using it we can create a directory junction[331], hard link[332] or soft/symbolic link[333].

Lastly, we can go over a reference implementation of "mklink" from ReactOS[334]. We can see the supported argument by "mklink" in the screenshot below[335].



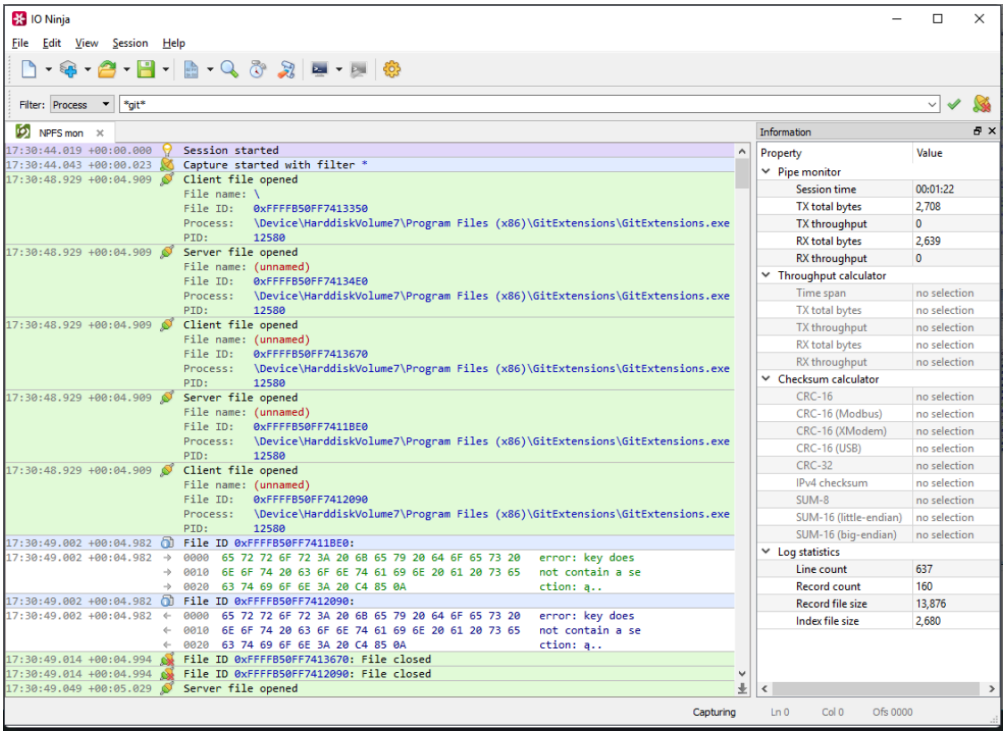
[330] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b
[331] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-junctions-bdbeb9aec611
[332] https://medium.com/@boutnaru/the-windows-concept-journey-8bc061672fe7
[333] https://medium.com/@boutnaru/the-windows-concept-journey-ntfs-symbolic-links-symlinks-127a1fbf4cc0
[334] https://github.com/reactos/reactos/blob/master/base/shell/cmd/mklink.c
[335] https://www.jamescoyle.net/how-to/1854-windows-has-symbolic-links-and-its-called-mklink

Pipes (Interprocess Communication)

In general pipes are an IPC (Inter-processes Communication) mechanism[336]. On Windows it is based on a section of shared memory. The process which created the pipe is called the "pipe server", while the process that connects to the pipe is called "pipe client". By the way, one process is writing to the pipe while the second process is reading from the pipe[337].

Overall, we have two types of pipes which are supported in Windows: "Anonymous Pipes" and "Named Pipes". The first has less overhead but also has limited functionality - more on each type in future writeups. In general, pipes can also be one way or two-way (duplex) which is also based on the type of the pipe[338].

Lastly, due to their rich functionality most of the time someone speaking about pipes they are probably talking about "Named Pipes". "Anonymous Pipes" are mostly used for child-parent communication. A simple example for an "Anonymous Pipe" is running "tasklist | findstr /i cmd.exe"[339]. There is no easy\builtin way to sniff pipe communication, however we use IO Ninja's "Pipe Monitor" for that - as shown in the screenshot below[340].



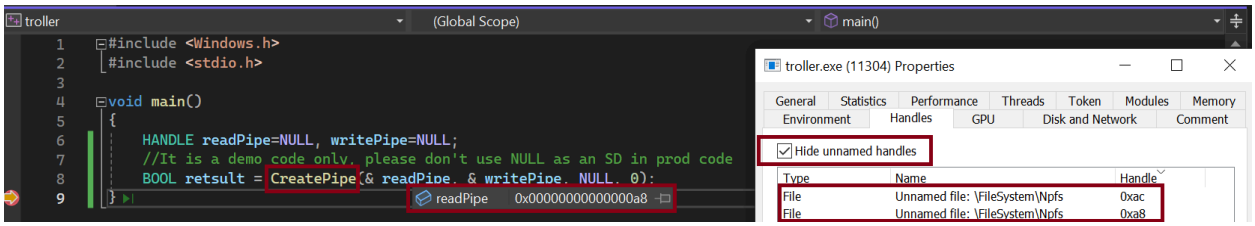
[336] https://medium.com/@boutnaru/windows-ipc-inter-process-communication-introduction-434c9287279b
[337] https://learn.microsoft.com/en-us/windows/win32/ipc/pipes
[338] https://learn.microsoft.com/en-us/windows/win32/ipc/about-pipes
[339] https://csandker.io/2021/01/10/Offensive-Windows-IPC-1-NamedPipes.html
[340] https://ioninja.com/plugins/pipe-monitor.html

Anonymous Pipes

An “Anonymous Pipe” is a type of pipe[341]. It supports only one-way communication, due to the fact that it is used mostly for transferring data between parent and child processes. Also, anonymous pipes are always local and cannot be used to transfer data over the network[342].

Overall, in order to create an anonymous pipe we can use the “CreatePipe” API call. The function returns handles to the read and write ends of the created pipe[343]. Also, a process can duplicate an handle to a pipe using the “DuplicateHandle”[344] API call and send it to a different process. We can use the “ReadFile”/”WriteFile” API for reading/writing to the pipe. An anonymous pipe exists until all the handles of the pipe (read/write) have been closed. This can be done using the “CloseHandle” API[345] or when the process terminates

Lastly, the usage of asynchronous (overlapped) read\write operations are not supported by anonymous pipes, due to that we can’t leverage the usage of “ReadFileEx” and “WriteFileEx”. Behind the scenes an anonymous pipe is implemented using a named pipe with a unique name. Thus, we sometimes can pass a handle to an anonymous pipe to a function that requires a handle to a named pipe - as shown in the screenshot below taken from both Visual Studio and Process Hacker[346].



.

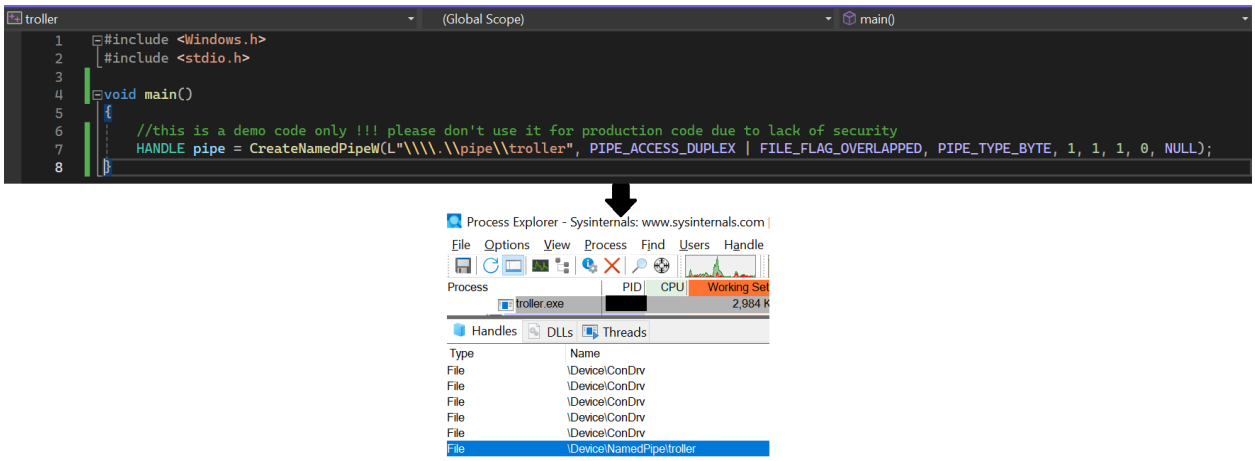
[341] https://medium.com/@boutnaru/the-windows-concept-journey-pipes-b5f59eaa561f
[342] https://learn.microsoft.com/en-us/windows/win32/ipc/anonymous-pipes
[343] https://learn.microsoft.com/en-us/windows/win32/api/namedpipeapi/nf-namedpipeapi-createpipe
[344] https://learn.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-duplicatehandle
[345] https://learn.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-closehandle
[346] https://learn.microsoft.com/en-us/windows/win32/ipc/anonymous-pipe-operations

Named Pipes

A “Named Pipe” is a type of pipe[347]. It supports both one-way or duplex pipe communication between a pipe server and one/more pipe clients. Also, named pipe provides communication between two (or more) processes on the same computer or on different computers across the network[348].

Moreover, named pipes can be protected using a “Security Descriptor”[349]. For creating a “Named Pipe” we can leverage the Win32 “CreateNamedPipeA”/“CreateNamedPipeW” API function[350], by providing a name and different attributes - as shown in the screenshot below. The process which creates the pipe is called the “Pipe Server”. For accepting connections the “Pipe Server” uses the “ConnectNamedPipe” Win32 API function[351].

Lastly, the “Pipe Client” connected to a “Named Pipe” using the “CreateFileA”/”CreateFileW”[352] function or the “CallNamedPipeA”/”CallNamedPipeW” Win32 API function call[353]. By the way, a specific process can be a “Pipe Server”/”Pipe Client”/both.



[347] https://medium.com/@boutnaru/the-windows-concept-journey-pipes-b5f59eaa561f
[348] https://learn.microsoft.com/en-us/windows/win32/ipc/named-pipes
[349] https://medium.com/@boutnaru/windows-security-security-descriptor-sd-ba95b8fa048a
[350] https://learn.microsoft.com/en-us/windows/win32/api/namedpipeapi/nf-namedpipeapi-createnamedpipew
[351] https://learn.microsoft.com/en-us/windows/win32/api/namedpipeapi/nf-namedpipeapi-connectnamedpipe
[352] https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilew
[353] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-callnamedpipea

SEH (Structured Exception Handling)

SEH (Structured Exception Handling) is a Microsoft extension for C\C++ that is used for handling specific exception code situations (like hardware faults). By leveraging SEH we can ensure that even in case of errors resources like memory/files are released[354]. We can see SEH is how Windows implements "try-except" statements - a code snippet of using SEH is shown in the screenshot below[355] .

Overall, we can think about SEH as Win32 equivalent to Unix signals[356]. In contrast to exceptions in C++, SEH is not typed. All exceptions in the case of SEH share the same data structure: an exception code (the cause), information on what code faulted and the content of the CPU registers held at the time of the fault[357].

Thus, we can retrieve a code that identifies the type of exception that occurs using the "GetExceptionCode" macro[358]. For retrieving a description of an exception and information about the computer state that exists for the thread when the exception occurs we can use the "GetExceptionInformation" macro[359].

Lastly, SEH is supported not only in user-mode of Windows but also in kernel mode[360]. We can see examples of using SEH in the source code of ReactOS[361]. Also, we can see SEH used in user-mode open source code from Microsoft[362].

```
1    // . . .
2    __try {
3        // guarded code
4    }
5    __except ( /* filter expression */ ) {
6        // termination code
7    }
8    // . . .
```

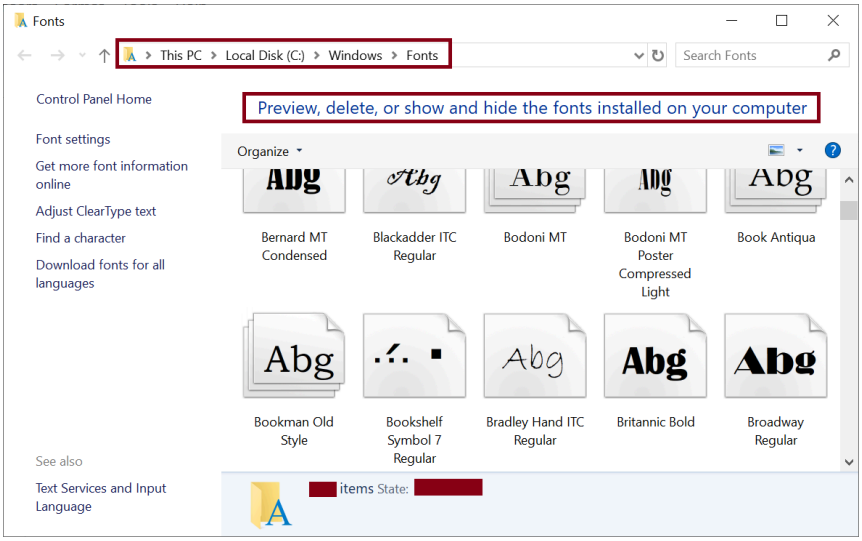
[354] https://learn.microsoft.com/en-us/cpp/cpp/structured-exception-handling-c-cpp
[355] https://richard-ac.github.io/posts/SEH/
[356] https://medium.com/@boutnaru/the-linux-concept-journey-signals-d1f37a9d2854
[357] https://stackoverflow.com/questions/2782915/what-should-i-know-about-structured-exceptions-seh-in-c
[358] https://learn.microsoft.com/en-us/windows/win32/debug/getexceptioncode
[359] https://learn.microsoft.com/en-us/windows/win32/debug/getexceptioninformation
[360] https://www.osronline.com/article.cfm%5Earticle=469.htm
[361] https://github.com/reactos/reactos/blob/master/modules/rostests/apitests/compiler/ms/seh/sehframes.cpp#L46
[362] https://github.com/microsoft/winget-cli/blob/master/src/Xlang/UndockedRegFreeWinRT/src/UndockedRegFreeWinRT/detours/modules.cpp#L266

%windir%\Fonts (Fonts Directory)

In general, fonts are stored as files. In the case of the Windows operating system, fonts are stored in "%windir%\Fonts" (like "C:\Windows\Fonts") - as shown below. Thus, we can install fonts just by dragging font files into that specific folder. We can also access that from "Control Panel-> Fonts" or "Control Panel-> Appearance and Personalization-> Fonts"[363].

Overall, we can also see (from the screenshot below) that the appearance of the font directory is customized. This behavior is managed by "fontext.dll" based on the usage of the "BD84B380-8CA2-1069-AB1D-08000948F534" CLSID as part of "desktop.ini". For a reference implementation of "fontext.dll" we can check out the source code of ReactOS[364].

Lastly, by checking the different file types stored on the fonts folder we can identify "*.ttf", "*.fon" and "*.ttc". TTF files are based on the "TrueType" specification which was initially created for macOS and later adapted by Windows[365]. TTC files are "TrueType Collections", this format can combine multiple font files into one file[366]. Fon files are based on the PE format which are used as a Windows font library[367].



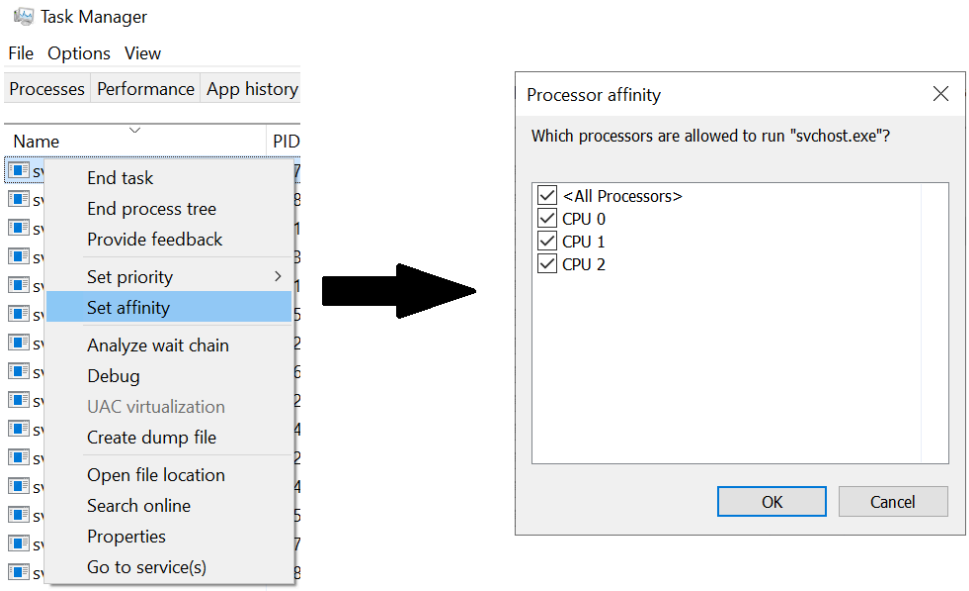
[363] https://support.microsoft.com/en-us/office/add-a-font-b7c5f17c-4426-4b53-967f-455339c564c1
[364] https://github.com/reactos/reactos/tree/master/dll/shellext/fontext
[365] https://docs.fileformat.com/font/ttf/
[366] https://docs.fileformat.com/font/ttc/
[367] https://docs.fileformat.com/font/fon/

Affinity (aka Affinity Mask)

The goal of a "Affinity" (aka "Affinity Mask") is to force Windows to schedule/use specific processors (logical processors to be precise) in order to execute threads of a specific process. By default, Windows puts the thread/s of an application on the least-busy processor. Thus, probably the only reason of setting an application to a single processor is in the case it does not work correctly when running on a multiprocessor system[368].

Overall, we can set the "Affinity Mask" for all the threads of a specified process using the "Task Manager" (taskmgr.exe) - as shown in the screenshot below. We can go over a reference implementation of this feature as part of RactOS, which is a free Windows-compatible Operating System[369].

Lastly, an "Affinity Mask" can be configured not only per process but also on a per thread basis. The first one can be done using the Windows API function "SetProcessAffinityMask"[370]. The second one can be done using the Windows API function "SetThreadAffinityMask"[371]. By the way, the "Affinity Mask" is a bit vector which represents a logical processor that a specific thread can be allowed to run on (in case of a process mask all threads of the process are allowed to run on the logical processor).



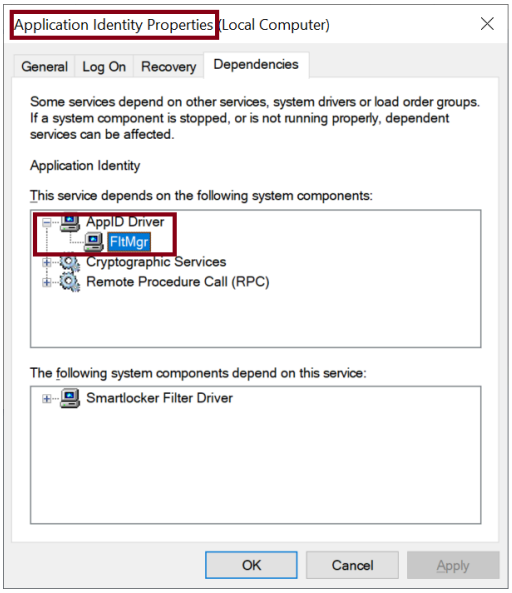
[368] https://superuser.com/questions/181577/what-is-windows-priority-and-affinity-and-what-advatanges-does-it-provide
[369] https://github.com/reactos/reactos/blob/master/base/applications/taskmgr/affinity.c
[370] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-setprocessaffinitymask
[371] https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-setthreadaffinitymask

AppIDSvc (Application Identity Service)

AppIDSvc (Application Identity Service) is a Windows service hosted by "svchost.exe"[372]. The description of the service states it is used for determining and verifying the identity of an application. Thus, if we disable this service it will prevent AppLocker[373] from being enforced.

Moreover, the service is launched using the access permissions of the "Local Service" user[374]. Also, the process of the service is defined with the protection level of "PsProtectedSignerWindows-Light", this is also known as "Protected Process Light" (PPL). It is important to understand that the actual protection level is a combination of "Type" and "Signer"[375].

Lastly, the "Application Identity Service" is dependent on: RPC ("Remote Procedure Call), "Cryptographic Services" (think about verifying digital signatures of executables) and the "AppID Driver" (%windir%\system32\drivers\appid.sys), which itself is dependent on "FltMgr" (%windir%\system32\drivers\fltmgr.sys) - as shown in the screenshot below. The user-mode part of the service is implemented as part of "%windir%\System32\appidsvc.dll", which is digitally signed by Microsoft.



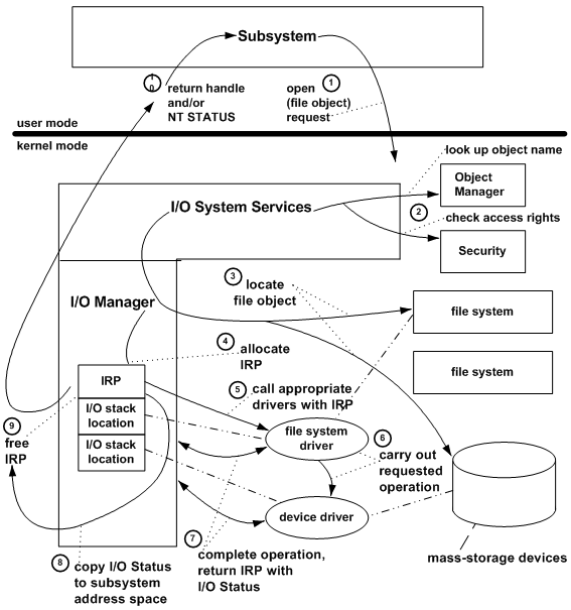
[372] https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f
[373] https://medium.com/@boutnaru/the-windows-security-journey-applocker-application-locking-b9547fb9cbbd
[374] https://medium.com/@boutnaru/the-windows-security-journey-local-service-nt-authority-local-service-b1a624472931
[375] https://uberagent.com/blog/uberagent-7-preview-edr-antivirus-windows-defender-protected-process-performance-monitoring/

IRP (I/O Request Packet)

On Windows most of the requests which are sent to device drivers are packaged as an IRQ (I/O Request Packet) - as shown in the diagram below[376]. An IRP is sent to a driver by a system component/driver by using the "IoCallDriver" macro that gets two parameters: pointer to a "DEVICE_OBJECT" and a pointer to IRP[377].

Overall, there are three most common types of IRPs. Write requests, which pass data to a driver so it can be written to a device. Read request, which passes a buffer to a driver so it can be filled with data from the device. Device I/O control (ioctl), which is used to communicate with drivers for any purpose which is not read/write[378].

Lastly, an IRP contains mainly details such as: the caller, parameters, state and more[379]. Every driver's I/O stack location (IO_STACK_LOCATION) contains a major function code (IRP_MJ_XXX). By using the specific code the driver/device knows which operations it needs to do regarding the IRP. Thus, each kernel driver must set the dispatch routines for the major function codes that it needs to provide[380] - more on that in future writeups.



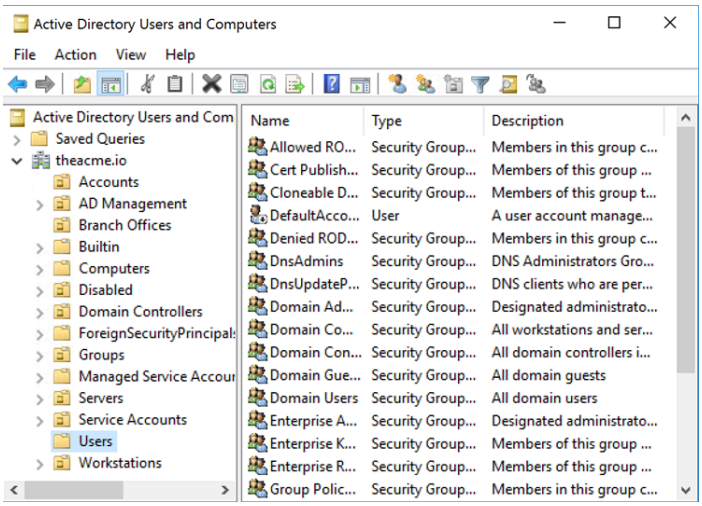
[376] https://www.easefilter.com/kb/understand-minifilter.htm
[377] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-iocalldriver
[378] https://www.windowsbugcheck.com/p/prerequisite-for-understanding-this.html
[379] https://rayanfam.com/topics/hypervisor-from-scratch-part-2/
[380] https://www.easefilter.com/kb/understand-irps.htm

AD (Active Directory)

AD (Active Directory) is an hierarchical database which is used for storing information about different objects on the network (users, computer, printers, shares and more). AD is composed of "schema" (defines the classes of objects and attributes stored by them), "global catalog" (contains a subset of attributes of every object in the directory), "query and index mechanism" and a "replication service" for distributing the data across the network[381]. Probably the most command UI for managing AD is the "Active Directory Users and Computers" snap-in (das.msc) - as shown in the screenshot below[382].

Overall, AD provides a security boundary and is used to federate authentication (aka "domain authentication" vs local SAM based) and policy enforcement. The AD database is stored on a DC (Domain Controller) inside the "ntds.dit" file - more on that in future writeups. By the way , there are multiple best practices (patching, monitoring, recovery plans and more) for security AD as part of Microsoft's documentation[383].

Lastly, due to the fact the AD database is stored on DCs we can access it remotely using the LDAP (Lightweight Directory Access Protocol) over TCP port 389 or 636 in case of LDAPS (LDAP over SSL). LDAP supports an interface of queries in order to ask the directory service for data[384] - more on that in future writeups.



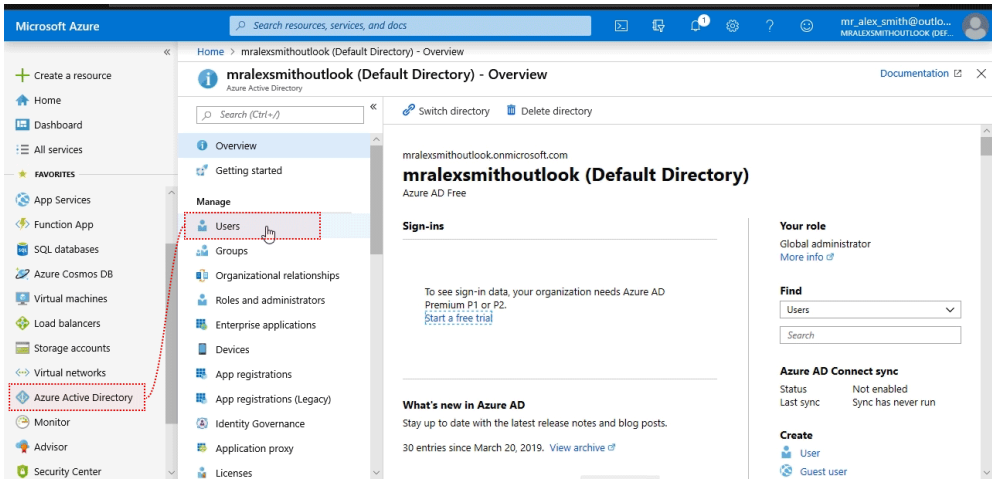
[381] https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview
[382] https://www.torchsec.org/what-is-azure-active-directory-active-directory-security/
[383] https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/security-best-practices/best-practices-for-securing-active-directory
[384] https://www.varonis.com/blog/the-difference-between-active-directory-and-ldap

AAD (Azure Active Directory)

AAD (Azure Active Directory) is a cloud directory and authentication service. It is leveraged by "Office 365" and "Azure" for account/groups/role management. AAZ can act as an IdP (Identity Provider) and supports federation (like SAML and more). As of 2023 AAD is deployed in over 30 datacenters around the world leveraging "Azure Availability Zones" present[385].

Overall, AAZ is not a cloud version of AD[386]. For example AAZ does not support NTLM and/or kerberos authentication methods and uses cloud based authentication mechanisms (like OAuth2, WS-Security and SAML). Also, LDAP is not used and REST (Representational State Transfer) API is used instead for communicating with web services. AAZ can provide mobile device management with Microsoft Intune while AD does not support mobile devices[387].

Lastly, for managing AAD we can use the "Microsoft Azure Portal" - as shown in the screenshot below[388]. By the way, on 2023 the name of "Azure Active Directory" was changed to "Microsoft Entra ID"[389] .



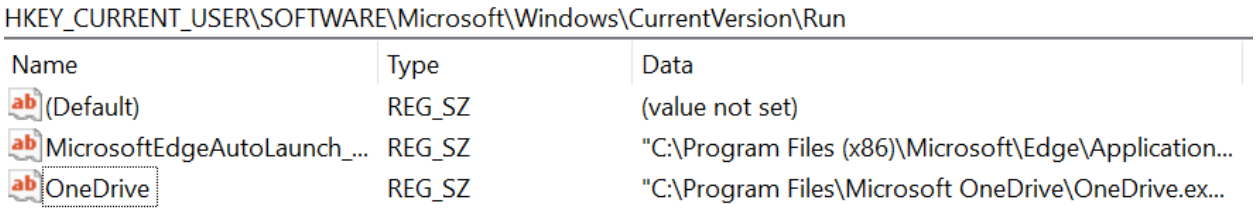
[385] https://www.torchsec.org/what-is-azure-active-directory-active-directory-security/
[386] https://medium.com/@boutnaru/the-windows-concept-journey-ad-active-directory-af9e795f86b4
[387] https://www.varonis.com/blog/azure-active-directory
[388] https://stealthmail.com/help/deployment-guide/add-user-to-azure-active-directory
[389] https://www.microsoft.com/en-us/security/blog/2023/07/11/microsoft-entra-expands-into-security-service-edge-and-azure-ad-becomes-microsoft-entra-id/

Run (Registry Key)

The registry has two relevant locations for the "Run" key[390]. Those are: "HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run" and "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run". The run registry key is used for configuring a list of programs\applications to execute when the user logs on.

Overall, each data value (as part of the run key) is the command line to execute (can also include arguments) and it is limited to 260 characters - as shown in the screenshot below. In case multiple programs\applications are registered the order of execution is indeterminate[391].

Lastly, entries as part of the "HKEY_LOCAL_MACHINE" run key will execute every time any user logs in to the system. On the other hand, entries part of the "HKEY_CURRENT_USER" run key are executed every time a specific user logs to the system[392].



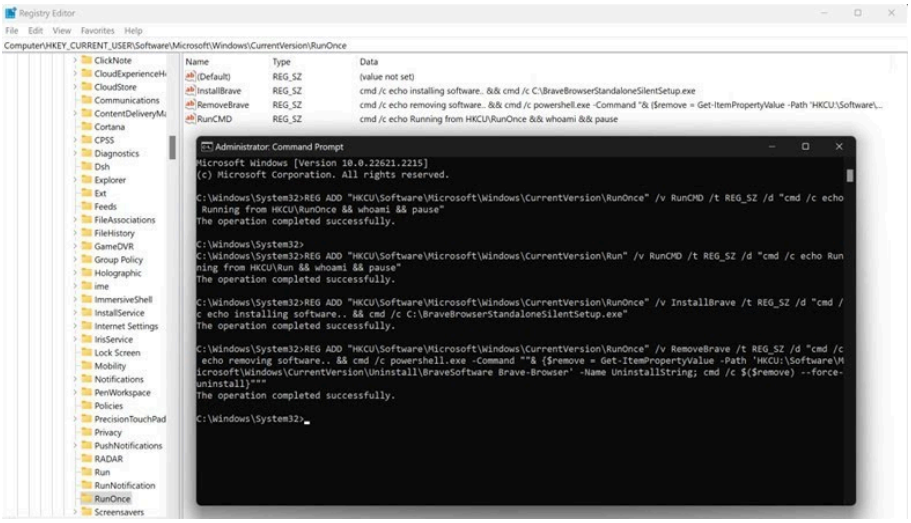
[390] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[391] https://learn.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys
[392] https://www.alkanesolutions.co.uk/2023/08/31/run-an-executable-after-windows-logon/

RunOnce (Registry Key)

The registry has two relevant locations for the "RunOnce" key[393]. Those are: "HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnce" and "HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce". The run registry key is used for configuring a list of programs\applications to execute when the user logs on - as shown in the screenshot below[394].

Overall, each data value (as part of the run key) is the command line to execute (can also include arguments) and it is limited to 260 characters - as shown in the screenshot below. In case multiple programs\applications are registered the order of execution is indeterminate[395].

Lastly, entries as part of the "HKEY_LOCAL_MACHINE" run key will execute once (the registry value will be deleted prior to the command line being run) for the first user (any user) that logs in to the system. On the other hand, entries part of the "HKEY_CURRENT_USER" run key are executed once when a specific user logs to the system. Also, because the command can fail to run (and won't run again) we can prefix the registry's value name with "!" which ensures the command is run successfully[396].



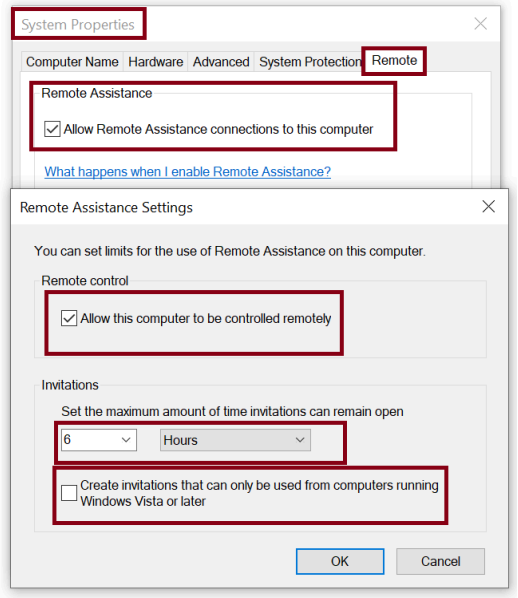
[393] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[394] https://www.youtube.com/watch?v=zgFzCq5uEPw
[395] https://learn.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys
[396] https://www.alkanesolutions.co.uk/2023/08/31/run-an-executable-after-windows-logon/

Remote Assistant

The Windows operating system provides a feature called “Remote Assistant” that can be used for requesting assistance from another Windows user over the Internet. The request is done by creating an encrypted invitation file and a password. The invitation is sent to a trusted party which uses that for connection and temporarily controlling the remote computer (in order to provide some kind of support). The controlled computer must have “Remote Assistance” and “Remote Desktop Connections” enabled[397].

Overall, “Remote Assistant” has been supported since Windows XP. The relevant registry[398] settings are located at “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Remote Assistance”. In order to enable remote assistant we should set “fAllowFullControl“ and “fAllowToGetHelp” to the value “1”[399].

Lastly, for security reasons we can set the maximum amount of time an invitation can remain open. Also, we can require that only computers using Windows Vista and later can use the invitation we create - as shown in the screenshot below. By the way, we can get to those configurations from the UI in the following manner: “Win+Pause (Break)”->”Remote Settings”[400].



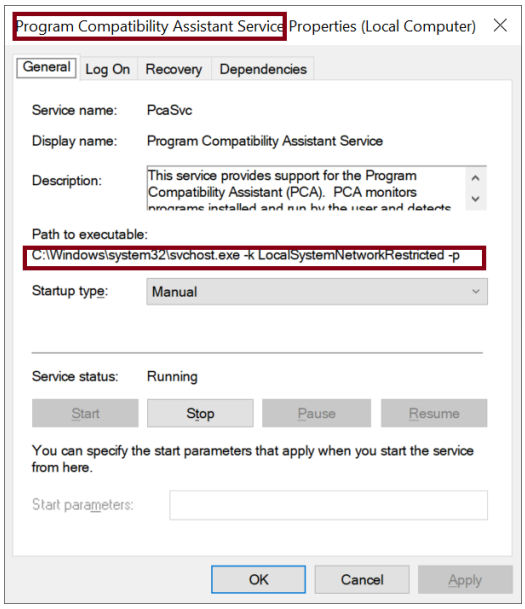
[397] http://www.ctimls.com/Support/KB/How%20To/Use_Windows_Remote_Assistance
[398] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[399] https://www.avica.com/blog/windows-remote-assistance/
[400] https://www.helpwire.app/blog/request-remote-assistance-windows-10/

PCA (Program Compatibility Assistant)

PCA (Program Compatibility Assistant) is a built-in feature of the Windows operating system which helps running older programs on new versions of Windows. It is implemented as a Windows service[401] called "PcaSvc" - as shown in the screenshot below.

Moreover, the service implementation is based on a DLL (%windir%\System32\pcasvc.dll, it is digitally signed by Microsoft) which is hosted by the "svchost.exe"[402] - also shown below. The description of the service states it monitors programs installed and run by the user and detects known compatibility problems. Also, If this service is stopped, PCA will not function properly.

Lastly, in case an application encounters compatibility issues, "PacSvc" may prompt the user with suggestions or automatically apply compatibility settings to ensure smooth execution. Thus, we can have automatic or user-initiated compatibility settings being applied to applications[403]. By the way, the service is executed with the permissions of the "Local System" user[404].



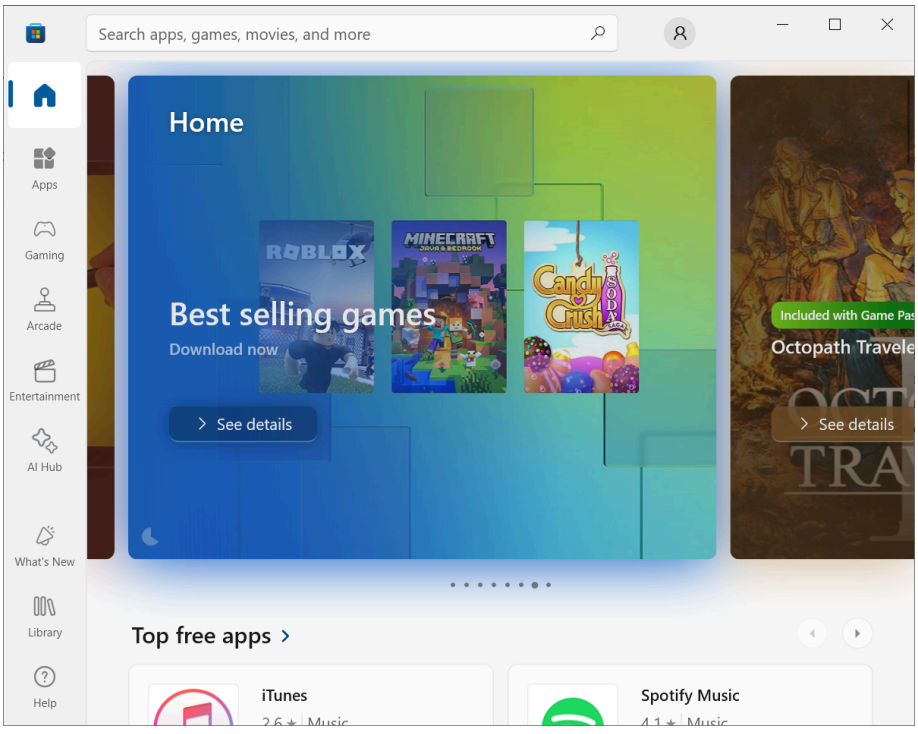
[401] https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4
[402] https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f
[403] https://malwaretips.com/blogs/program-compatibility-assistant-service/
[404] https://medium.com/@boutnaru/the-windows-security-journey-local-system-nt-authority-system-f087dc530588

Microsoft Store (formally Windows Store)

Microsoft Store was formerly known as Windows Store, it is a distribution platform created and managed by Microsoft. It was created in order to be an application marketplace for Windows 8, allowing the distribution of UWP (Universal Windows Platform) applications. Since Windows 10 (1803) different distribution platforms ("Xbox Store", "Xbox Video", "Xbox Music", "Windows Phone Store" and "Windows Market Place") were merged into "Microsoft Store"[405].

Moreover, we can think about "Microsoft Store" as an equivalent to Apple's "App Store" and "Google Play". We can access it using the "Microsoft Store" app (as shown in the screenshot below) or by using the following link: "https://apps.microsoft.com/". By using the Microsoft Store we can easily find/install/uninstall applications[406].

Lastly, users can purchase different items from the Microsoft Store. Examples of such items are: software (like Adobe Photoshop and Microsoft Teams), business applications (like Microsoft 365 and Dynamic 365), games, movies and TV shows, hardware, developer and IT tools[407].



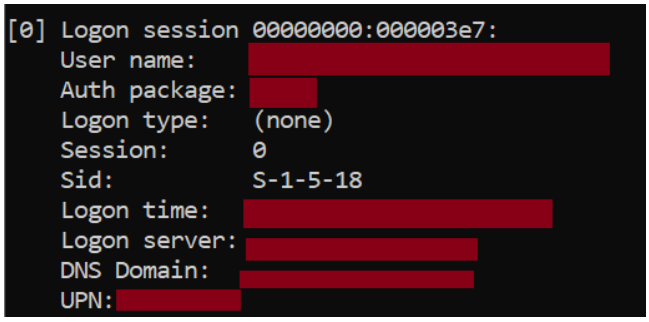
[405] https://en.wikipedia.org/wiki/Microsoft_Store
[406] https://www.pcmag.com/picks/best-apps-in-the-windows-11-store
[407] https://www.techtarget.com/searchmobilecomputing/definition/Windows-Store

LUID (Locally Unique Identifier)

LUID (Locally Unique Identifier) is a 64bit number which is created as unique until the system is restarted[408]. LUID is stored using the "struct _LUID" structure as part of "ntdef.h"[409].

Overall, we can create a LUID using functions like "AllocateLocallyUniqueId"[410]. Also, we can use other functions\macros for handling LUID such as the "RtlConvertLongToLuid" function which is used for converting a long integer to LUID[411] and "RtlEqualLuid" macro[412].

Lastly, LUID is used in different places as part of the Windows operating systems. For example LUID is by logon session IDs[413] - as shown in the screenshot below (taken using Sysinteranls' "logonsessions64.exe"). The abstract WMI class "Win32_LUID" represents a locally unique identifier. It is defined as part of the "%windir%\system32\wbem\wmipjobj.mof" MOF file and implemented as part of "%windir%\system32\wbem\WMIPJOBJ.dll"[414].



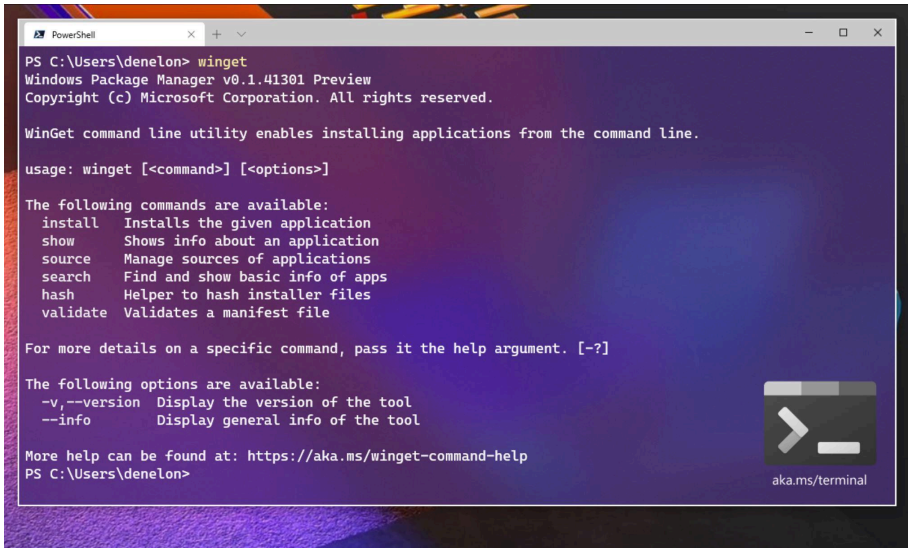
[408] https://learn.microsoft.com/en-us/windows/win32/secgloss/l-gly
[409] https://learn.microsoft.com/en-us/windows/win32/api/ntdef/ns-ntdef-luid
[410] https://learn.microsoft.com/en-us/windows/win32/api/securitybaseapi/nf-securitybaseapi-allocatelocallyuniqueid
[411] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/nf-ntddk-rtlconvertlongtoluid
[412] https://learn.microsoft.com/en-us/windows-hardware/drivers/ddi/wdm/nf-wdm-rtlequalluid
[413] https://www.microsoftpressstore.com/articles/article.aspx?p=2224373&seqNum=7
[414] https://learn.microsoft.com/en-us/previous-versions/windows/desktop/wmipjobobjprov/win32-luid

Windows Package Manager

In general, we can define a package manager as a set of tools used for installing/configuring/updating software in an automatic manner[415]. Even though we have the "Microsoft Store"[416] the "Windows Package Manager" is still very relevant[417]

Overall, the Windows package manager was included as part of Windows 11 and has been added to Windows 10 through an update to the operating system. By the way, there are also other package managers for Windows (which are not builtin) such as: Chocolatey, Scoop and Ninite[418].

Lastly, the "Windows Package Manager" is controlled using the "winget.exe" utility (more on that in a future writeup)- as shown in the screenshot below[419]. Thus, we can use it for searching/viewing/installing commonly used developer tools. It was announced as part of the "Microsoft Build Developer Conference 2020"[420].



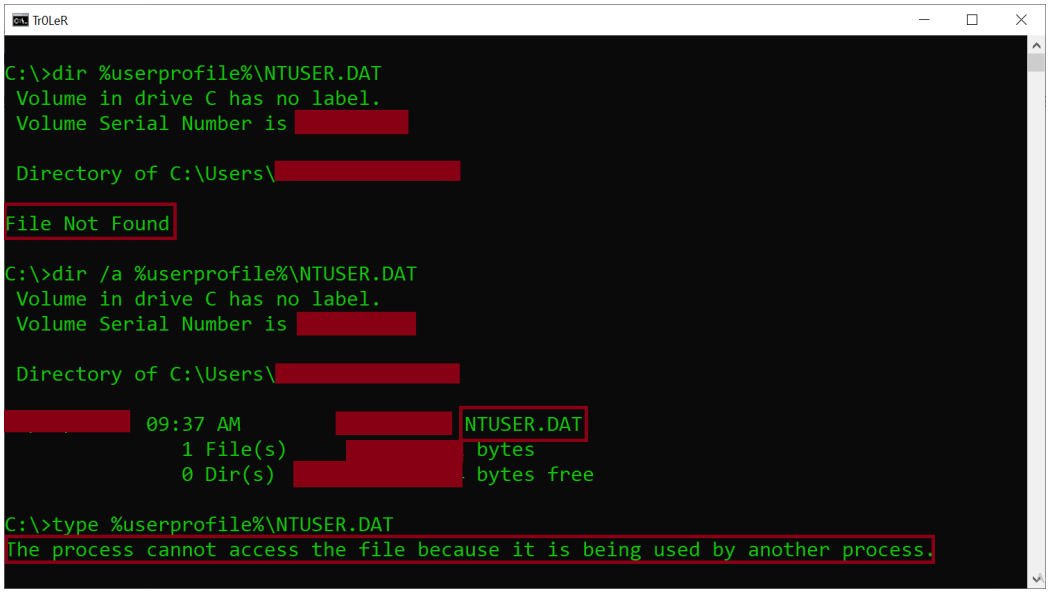
[415] https://learn.microsoft.com/en-us/training/modules/explore-windows-package-manager-tool/2-explain-purpose-of
[416] https://medium.com/@boutnaru/the-windows-concept-journey-microsoft-store-formally-windows-store-75d7f2d5370b
[417] https://www.windowscentral.com/software-apps/is-overreliance-on-copilot-chatgpt-making-you-dumber
[418] https://blog.logrocket.com/6-best-package-managers-windows-beyond/
[419] https://puresourcecode.com/news/microsoft-announces-windows-package-manager-a-new-way-to-install-tools-easily-on-windows/
[420] https://venturebeat.com/business/microsoft-windows-package-manager-powertoys/

NTUSER.DAT

The NTUSER.DAT contains user account settings and customizations of a specific Windows user (which can be a local user or a domain user), think about the wallpaper settings as an example or the preferred keyboard layout. It is created by the operating system the first time a user logs on the system. The file is located in the user profile directory of the user "%userprofile%\NTUSER.DAT"[421].

Overall, the file is hidden thus we can see it using the "/a" flag of "dir" which is a builtin command of cmd.exe[422] - as shown in the screenshot below. The "NTUSER.DAT" is basically a registry hive[423] which is loaded to "HKEY_USERS" and is pointed to by "HKEY_CURRENT_USER" when the user logs on to the system. We can use the "NTUSER.DAT" file for offline analysis on a non-running system.

Lastly, there are also backups and transaction logs for the "NTUSER.DATA" (also stored in the %userprofile% directory with extensions like ".log"). The "ntuser.ini" file describes roaming profiles used in networked environment[424]. As with the files of the system's registry ("%windir%\system32\config"), both "NTUSER.DAT" and its related files are opened exclusively by the operating system when the user is logged on.



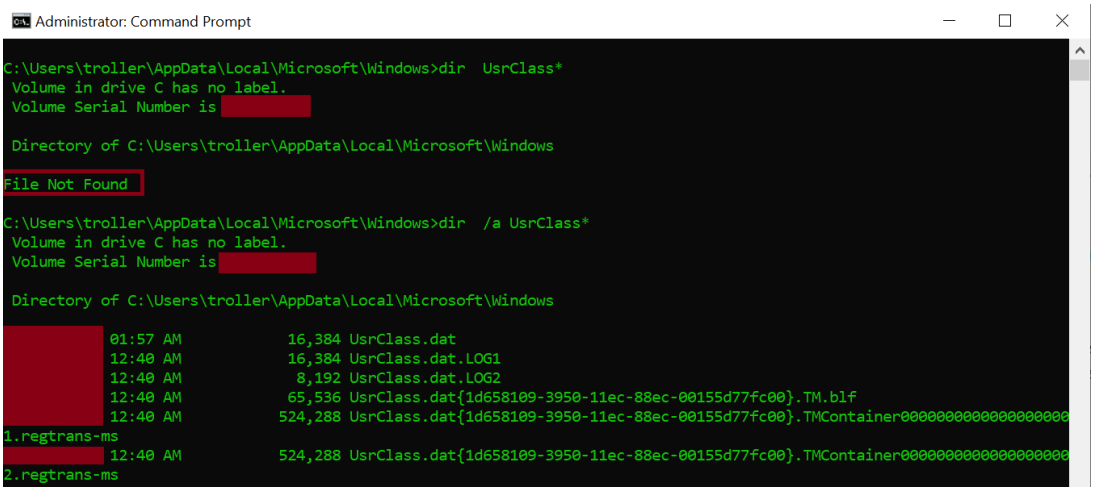
[421] https://appuals.com/ntuser-dat-file-explained/
[422] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b
[423] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[424] https://www.techtarget.com/searchenterprisedesktop/blog/Windows-Enterprise-Desktop/Understanding-NTUserdat-in-Windows-10

UsrClass.dat

The “UsrClass.dat” file is located “C:\Users\%username%\AppData\Local\Microsoft\Windows” (which can be accessed also by “%userprofile%\AppData\Local\Microsoft\Windows”). “UsrClass.dat” is a registry hive file. Together with the “NTUSER.DAT”[425] registry hive file it composes the registry hive of a logged on user (created under HKEY_USERS and pointed by HKEY_CURRENT_USER).

Moreover, like with “NTUSER.DAT” there are backups\transactions files. By the way, the “UsrClass.dat” is also a hidden file - as shown in the screenshot below. In case we create a new local user account and perform a secondary logon using “runas.exe”[426] a “UsrClass.dat” file is created. The newly created registry file contains default subkeys like “CLSID” and “Local Settings”.

Lastly, “UsrClass.dat” contains highly valuable forensics artifacts like “ShellBags”. Those are records of the user’s view settings and preferences while exploring folders[427].



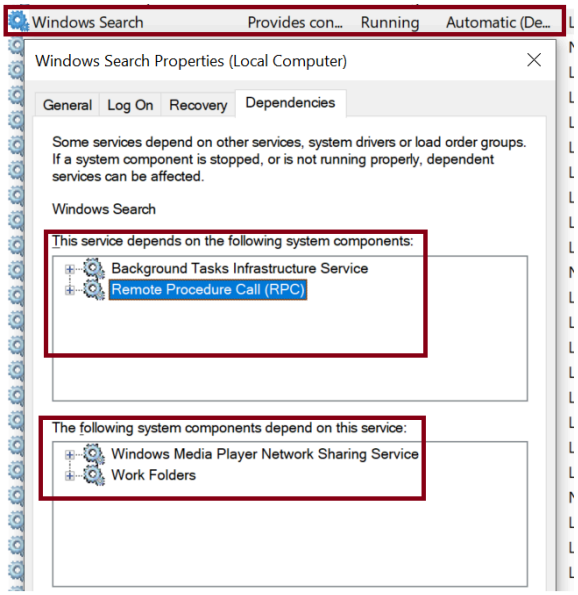
[425] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[426] https://medium.com/@boutnaru/the-windows-process-journey-runas-exe-run-as-utility-3c1e0b8aaa67
[427] https://forensafe.com/blogs/shellbags.html

Windows Search

"Windows Search" is an operating system service[428] - as shown in the screenshot below. Based on its description it is used for content indexing, property caching, and search results for files, e-mail, and other content. The service name is "WSearch" and the executable of the service is located at "%windir%\system32\SearchIndexer.exe". Also, the binary is executed while passing "/Embedding" as a command line argument.

Overall, by default the service is started automatically when the operating system boots up. Also, the service is executed with the permissions\rights of the "Local System"[429] user.

Lastly, the "Windows Search" service is dependent on the "Background Task Infrastructure Service" and "Remote Procedure Call" service. Moreover, the "Windows Media Player Network Sharing Service" and the "Work Folders" service are dependent on the "Windows Search" service.



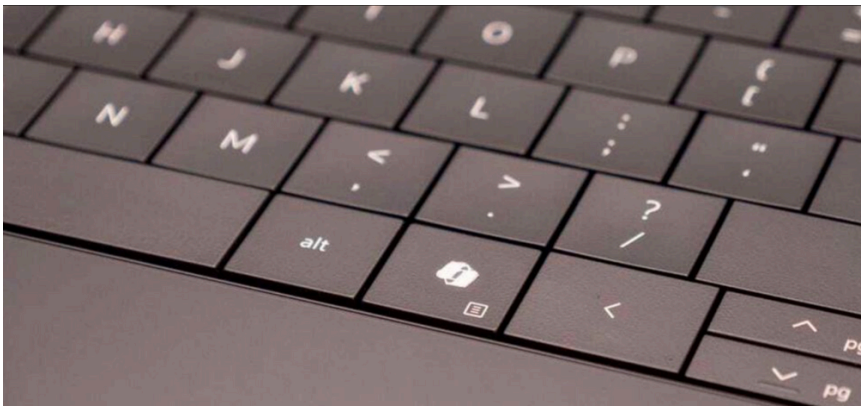
[428] https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4
[429] https://medium.com/@boutnaru/the-windows-security-journey-local-system-nt-authority-system-f087dc530588

Windows Copilot

“Windows Copilot” is a conversational chat interface that lets the user perform different tasks which includes: searching for specific information, generating text (summaries/email/etc.), creating images based on prompts, writing code and more. We can access “Copilot” over the web[430] or we can access it from the sidebar in Windows[431]. We can also use it from “Bing Search”/”Bing Mobile App”, the sidebar in the Edge browser, the “Copilot Mobile App” for Android/iOS or “Microsoft 365 Apps” (if we have a paid subscription).

Overall, Copilot has some distinguishing features from other GenAI (Generative AI) chatbots such as: image upload, voice input and spoken responses, choice of style, link to sources, choices of AI models and image generation for free[432].

Lastly, starting for 2024 a dedicated “Copilot Key” was announced as part of Windows based keyboards layouts - as shown in the image below[433]. In general, is based on OpenAI’s GTP-4 LLM which had been fine tuned [434].



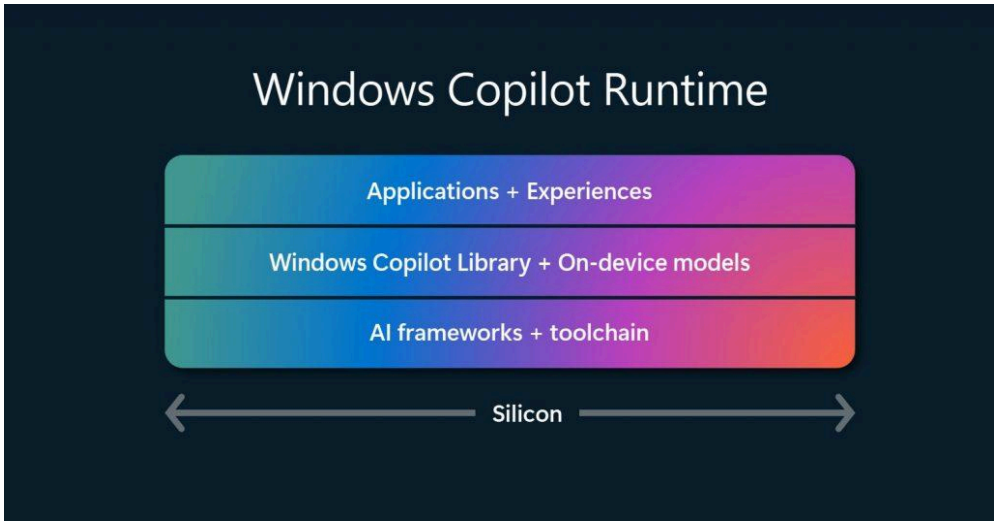
[430] https://copilot.microsoft.com/
[431] https://en.wikipedia.org/wiki/Microsoft_Copilot
[432] https://www.pcmag.com/explainers/what-is-microsoft-copilot
[433] https://trak.in/stories/microsofts-pc-keyword-gets-1st-change-in-30-years-a-new-button-for-ai-called-copilot-key/
[434] https://support.microsoft.com/en-au/topic/chatgpt-vs-microsoft-copilot-what-s-the-difference-8fdec864-72b1-46e1-afcb-8c12280d712f

Windows Copilot Runtime

The goal of "Windows Copilot Runtime" is to provide a way of iterating with the installed operating system using AI. "Microsoft Research" has created an SLM (Small Language Model) which can provide many of the same capabilities/features found in a LLM (Large Language Module). However, the SLM is more compact and efficient which allows it to run locally on Windows[435]. We can think about it as a end-to-end Windows ecosystem - as shown in the diagram below[436].

Overall, the "Windows Copilot Runtime" provides AI backed APIs which are also known as "Windows Copilot Library". By using those APIs a developer can run/find/optimize their own ML (Machine Learning) model. Examples of AI-backed APIs are: Phi Silica, Text Recognition with OCR, Recall, and Studio Effects[437].

Lastly, there are many examples of using "Windows Copilot Runtime" for integrating AI into Windows applications. I recommend checking out Microsoft's "AI on Windows Sample Gallery"[438]. Also, a good read is the FAQs (Frequently Asked Questions) about using AI with Windows[439]. There are also guidelines for responsible development of generative AI apps and features on Windows[440].



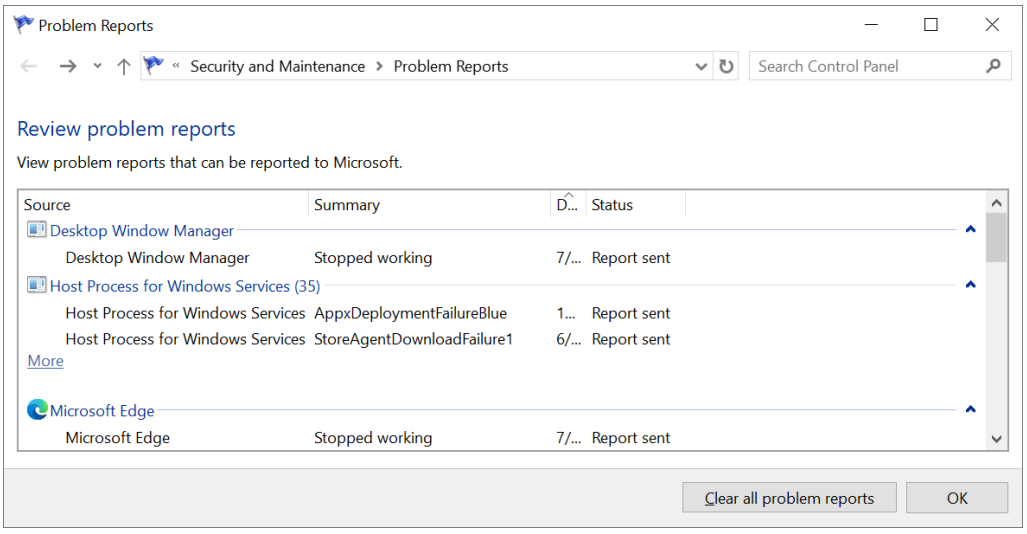
[435] https://learn.microsoft.com/en-us/windows/ai/overview
[436] https://blogs.windows.com/windowsdeveloper/2024/05/21/unlock-a-new-era-of-innovation-with-windows-copilot-runtime-and-copilot-pcs/
[437] https://learn.microsoft.com/en-us/windows/ai/apis/
[438] https://learn.microsoft.com/en-us/windows/ai/samples/
[439] https://learn.microsoft.com/en-us/windows/ai/faq
[440] https://learn.microsoft.com/en-us/windows/ai/rai

WER (Windows Error Reporting)

The goal of WER ("Windows Error Reporting") is to provide users with the ability to notify Microsoft in case of such as kernel faults/unresponsive applications/application faults. Microsoft can use the reports sent for providing customers with troubleshooting information and/or solutions. Users can enable WER manually or it can be enabled by an administrator using group policy[441].

Moreover, WER supports five different operation modes. Headless reporting which allows collecting reports which out interfering the users (this option can only be used with corporate reporting). Corporate reporting allows sending all data to a file share instead of uploading it to Microsoft (it can't be used with internet reporting mode). Internet reporting mode sends all data to Microsoft. Shared memory reporting is in case the application security context is the same as the logged on user, thus the error reporting stem can use a block of shared memory for communication. In case the security context is different a file is used for communication in a manifest reporting mode[442].

Lastly, information regarding the WER error codes, functions, settings, data structures and enums can be found as part of the Microsoft documentation[443]. By the way, applications can save their state/data before exiting due to an unhandled exception or when the application stops responding. There is also an option of restarting the application[444]. We can view the reports sent to Microsoft from the "Control Panel"[445] in the following location "System and Security\Security and Maintenance\Problem Reports" - as shown in the screenshot below.



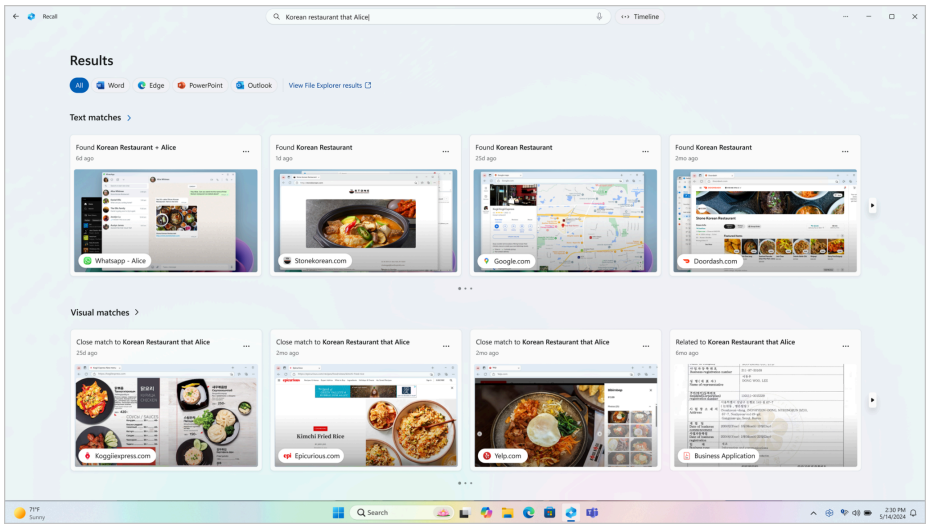
[441] https://learn.microsoft.com/en-us/windows/win32/wer/windows-error-reporting
[442] https://learn.microsoft.com/en-us/windows/win32/wer/using-wer
[443] https://learn.microsoft.com/en-us/windows/win32/wer/wer-reference
[444] https://learn.microsoft.com/en-us/windows/win32/recovery/application-recovery-and-restart-portal
[445] https://medium.com/@boutnaru/the-windows-concept-journey-control-panel-34bf84ca7ff0

Windows Recall

The purpose of "Windows Recall" is to allow users to retrace things that they have done on a specific Windows system. By using recall the operating system provides an explorable timeline of the user actions. Thus, we just need to describe how we remember what we want to retrace and Recall will take us to that point in time- as shown in the screenshot below. This is done by taking a snapshot (stored locally) every 5 seconds (while the screen content is changed)[446].

Moreover, "Windows Recall" has the following minimum system requirements: 8 logical processors, 16 GB RAM and at least 50 GB of space for enabling recall (256 GB is recommended). The last requirement is "Copilot + PC", that is a new class of Windows 11 system that is powered by a turbocharged NPU (neural processing unit) – a computer chip for AI-intensive processes[447].

Lastly, we can control which applications we exclude from recall (think about banking apps\websites). Of course it is relevant only for supported browsers (in regards to websites), we can add such filters in "Windows Settings > Privacy & Security > Recall & Snapshots" and click on "Add website" or "Add App" defending on what we want to exclude[448]. We can access the recall feature using "WinKey+J" key combination or by a dedicated key if we have it in our keyboard.



[446] https://support.microsoft.com/en-us/windows/retrace-your-steps-with-recall-aa03f8a0-a78b-4b3e-b0a1-2eb8ac48701c
[447] https://www.microsoft.com/en-gb/windows/copilot-plus-pcs#faq1
[448] https://pureinfotech.com/exclude-apps-websites-recall-windows-11/

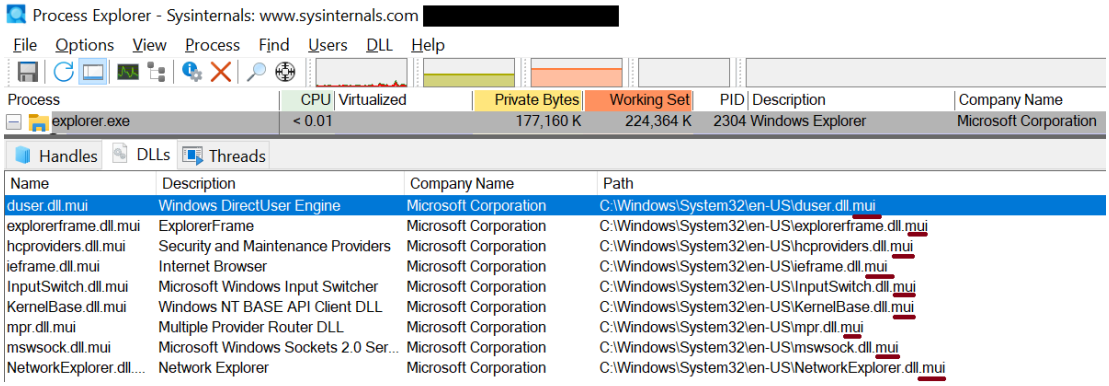
Multilingual User Interface (MUI)

Multilingual User Interface (MUI) is a technology that is used for enabling multilingual user experiences. It is needed to help Microsoft benefit from the growth opportunity in international markets. As of 2021 more than 91.5% of the world population are non-English speakers. At the end MUI provides benefits for developers, enterprises and OEMs[449].

Moreover, MUI was introduced as part of Windows 2000. "*.mui" files hold resources that allow the Windows interface to display different languages. The operating system opens those files based on instructions given by the user (based on configuration). By doing so we can change languages without replacing the binaries of the OS[450].

Overall, the main fundamental concepts of MUI are: separation of source code from language specific resources, dynamically loading language-specific resources and building MUI apps[451]. For loading an MRU library we can use "LoadMUILibraryA"[452] or "LoadMUILibraryW" and it will be loaded and an handle returned[453] - we can see an example in the printscreen below. For unloading we can use "FreeMUILibrary"[454].

Lastly, the format used by MUI files is PE[455] and they are loaded by executables that support MUI as shown in the screenshot below. The default OS MUI files are stored in "%windir%\system32" in directories that represent the different languages such as "en-US","en-GB", "fr-CA" and "fr-FR".



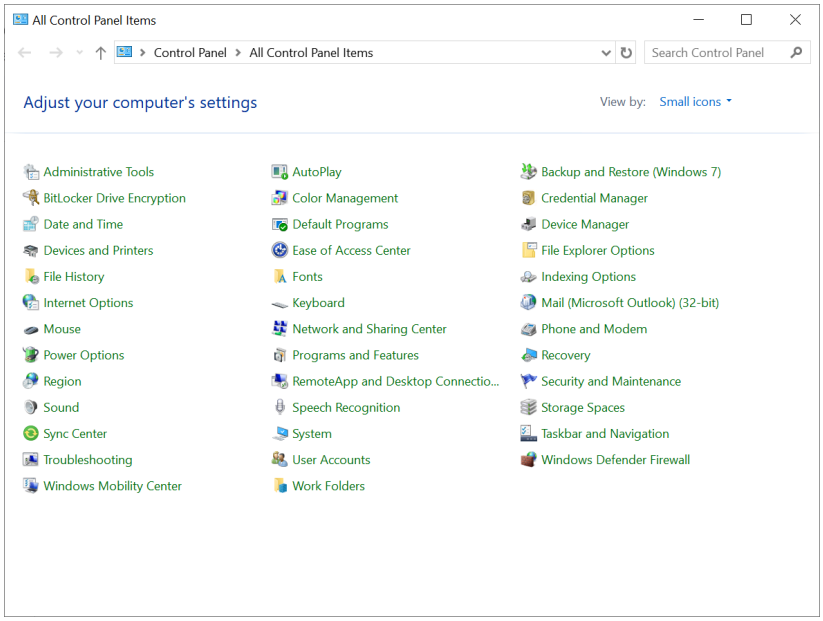
[449] https://learn.microsoft.com/en-us/windows/win32/intl/overview-of-mui
[450] https://openmuifile.com/muicache.html
[451] https://learn.microsoft.com/en-us/windows/win32/intl/mui-fundamental-concepts-explained
[452] https://learn.microsoft.com/en-us/windows/win32/api/muiload/nf-muiload-loadmuilibrarya
[453] https://learn.microsoft.com/en-us/windows/win32/api/muiload/nf-muiload-loadmuilibraryw
[454] https://learn.microsoft.com/en-us/windows/win32/api/muiload/nf-muiload-freemuilibrary
[455] https://medium.com/@boutnaru/the-portable-executable-journey-dos-header-ea5b29f15612

Control Panel

The goal of the "Control Panel" is to help users with configuring system-level features of the operating system. Examples for those are: system maintenance, security, hardware/software setup and user account management. When speaking about "Control Panel" we usually mean the entire Windows control panel feature, while specific control panels are referred by "Control Panel Items"[456] - as shown in the screenshot below.

Overall, the control panel's items are also called "applets". Each applet is a "*.CPL" file which is basically a DLL/PE file that exports the "CPlApplet" function. There are a couple of ways for registering an "applet". Placing the "*.CPL" file in the "%windir%\System32" directory. Also, adding the information (location/name of the CPL file) about the applet in the following subkey: "HKLM\Software\Microsoft\Windows\CurrentVersion\Control Panel\Cpls"[457].

Lastly, we can also add the CLSID (class identifier, part of Microsoft's "Component Object Model" aka COM) for the applets in the following location: "HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\ControlPanel\NameSpace\" [458]. By the way, the configuration done by applets is mostly relevant for the local machine, as opposed to MMC snap-ins which are hosted by "mmc.exe"[459]. Those snap-ins also support remote management using protocols like MS-RPC.



[456] https://learn.microsoft.com/en-us/windows/win32/uxguide/winenv-ctrl-panels
[457] https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/hh127454(v=vs.85)
[458] https://winaero.com/how-to-add-anything-you-want-to-control-panel/
[459] https://medium.com/@boutnaru/the-windows-process-journey-mmc-exe-microsoft-management-console-a584afe66d86

Types of Windows Applications

There are different types of Windows applications, which we can execute on Windows based devices. Among those types we can find: “Microsoft 365 Applications”, “Power Apps”, “.NET Applications”, “Windows Applications”, “Web Applications” and “Android Applications”[460]

Overall, each type has its own advantages and disadvantages and thus specific use cases in which it is more relevant - an example for that is shown in the table below[461]. There are different deployment mechanisms that can be used. From manual installation to scalable remote deployment (and even using package managers) - more on that also in future writeups.

Lastly, I believe that everyone that wants to get a better understanding of “how Windows works under the hood?” should get a good understanding of the different types of Windows applications.

Feature	WinUI	MFC	WinForms	WPF	UWP XAML
Support	Supports desktop and UWP apps	Supports native C++ apps	Supports .NET apps	Supports .NET apps	Supports .NET and C++ apps
Usability	Can work in Win32/desktop apps and UWP apps	Only works in Win32/desktop apps	Only works in Win32/desktop apps	Only works in Win32/desktop apps	Only works in UWP apps
Category	Modern native UI platform for Windows	Desktop/Win 32 apps	Desktop/Win 32 apps	Desktop/Win 32 apps	UWP apps
C++ Compatibility	Compatible with C++	Compatible with C++	Not compatible with C++	Not compatible with C++	Compatible with C++
.NET Framework Compatibility	Compatible with .NET	Not compatible with .NET	Compatible with .NET	Compatible with .NET	Compatible with .NET
Built-in Fluent Design Controls and Styles	Embodies fluent design controls that give Windows apps a polished look and feel	Does not contain modern controls and styles for building Windows apps	Does not contain modern controls and styles for building Windows apps	Does not contain modern controls and styles for building Windows apps	Has fluent design controls and styles
Built-in Support for Modern Input	Built-in support for modern input such as touch, pen, and gamepad	No built-in support for modern input	No built-in support for modern input	No built-in support for modern input	Built-in support for modern input

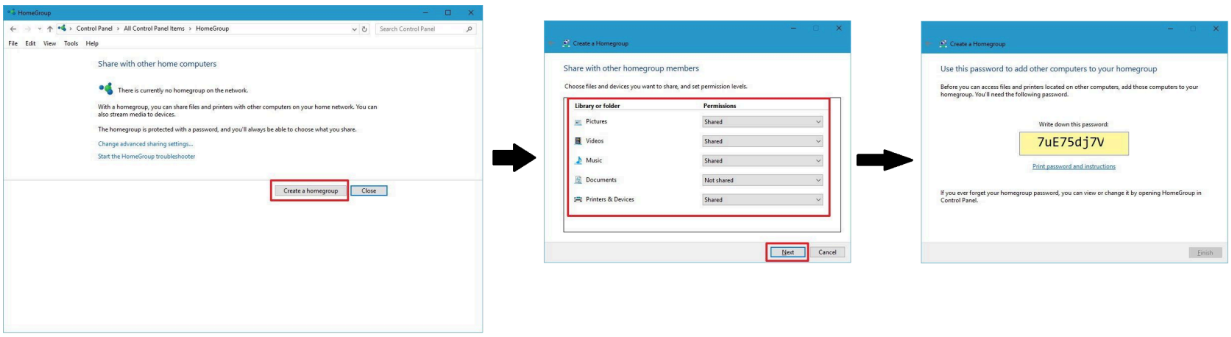
[460] https://learn.microsoft.com/en-us/windows/application-management/overview-windows-apps
[461] https://developer.mescius.com/blogs/winui-vs-wpf-winforms-uwp-and-mfc

Windows Homegroup

Basically, a homegroup is a group of PCs on a home network that can share resources between them (like printers and files). We can protect a home using a password (that we can modify anytime). After we create/join a homegroup we need to select the libraries we want to share (like "My Documents" and "My Pictures"). Homegroup has been introduced since Windows 7[462]. We can configure a homegroup on Windows 10 as shown below[463].

Overall, the homegroup feature has been removed from Windows 10 (Version 1803) . Thus, "Homegroups" won't appear as part of the "File Explorer" user interface[464] and as part of the "Control Panel" user interface[465] anymore. It is important to know that folders/files/printers that were previously shared using HomeGroup will continue to be shared. They can be accessed using "\\[PC]\[SharedFolder]"[466].

Lastly, although the "HomeGroup" feature is deprecated we can still share printers/files/folders using other features built into Windows 10 (Version 1803) and above[467].



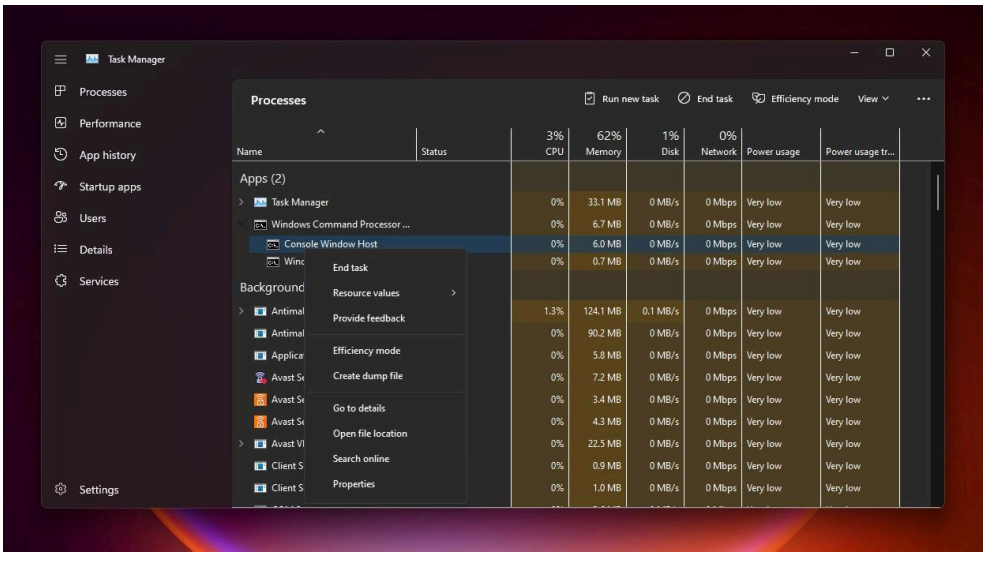
[462] https://support.microsoft.com/en-us/windows/homegroup-from-start-to-finish-9f802c8c-900f-60fb-826f-6fe06add8fe9
[463] https://www.windowscentral.com/how-setup-and-manage-windows-10-homegroup-local-network
[464] https://medium.com/@boutnaru/the-windows-concept-journey-file-explorer-previously-windows-explorer-e48077b135a0
[465] https://medium.com/@boutnaru/the-windows-concept-journey-control-panel-34bf84ca7ff0
[466] https://support.microsoft.com/en-us/windows/homegroup-removed-from-windows-10-version-1803-07ca5db1-7bca-4d11-68a3 -a31ff4a09979
[467] https://tinyurl.com/459u98aa

Task Manager

“Task Manager” is used in order to view/manage current running processes, view system resources usage, analyze performance and close unresponsive applications by leveraging its user interface[468]. The binary (%windir%\system32\Taskmgr.exe) is digitally signed by Microsoft. By the way, on 64-bit Windows systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\Taskmgr.exe”.

Overall, since Windows 11 22H2 “Task Manager” has a new design based on Fluent UI and WinUI. Thus, the classic interface was changed to a hamburger menu layout - as shown in the screenshot below. We can find the different viewing options: “Processes” (limited information about each running process) , “Performance” (CPU/memory/IO/networking usage and performance), “App History” (usage history for UWP applications), “Startup Apps”, “Users”, “Details” and “Services” on the hamburger menu in the left side of the UI. This has been done to improve the accessibility in case of touchscreen based devices[469].

Lastly, we can go over a reference implementation of “takmgr.exe” as part of ReactOS[470]. Also, there are different ways to open “Task Manager” such as (but not limited to): “CTRL+Shift+ESC”, “CTRL+ALT+DELETE”-> “Task Manager” and “WinKey+X”->”Task Manager”[471]. By the way, based on the command line arguments passed to “taskmgr.exe” we can identify the way in which it was launched[472].



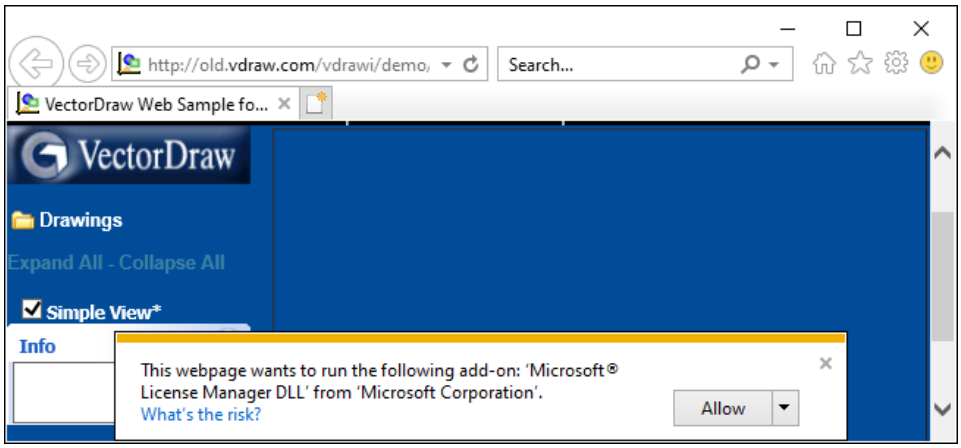
[468] https://www.spyshelter.com/exe/microsoft-windows-taskmgr-exe/
[469] https://www.bleepingcomputer.com/news/microsoft/hands-on-with-windows-11s-new-task-manager/
[470] https://github.com/reactos/reactos/tree/master/base/applications/taskmgr
[471] https://www.howtogeek.com/66622/stupid-geek-tricks-6-ways-to-open-windows-task-manager/
[472] https://www.hexacorn.com/blog/2018/07/22/taskmgr-exe-slashing-numbers/

ActiveX Controls

ActiveX controls are small applications which can be used by websites for providing contents such as videos\games. Because those applications can malfunction\damage the system there are security countermeasures such as : “ActiveX Filtering”, “digital signing” and “stringent default security settings”[473]. Popular Internet Explorer plug-ins like Adobe Flash, Adobe Shockwave, RealPlayer, Apple QuickTime, and Windows Media Player were implemented using ActiveX controls[474]

Overall, Microsoft introduced ActiveX in 1996. It is considered as a deprecated technology. Although ActiveX is not dependent on the Windows operating system most of them run on Windows and also require a x86 based CPU. OLE 2.0 and COM are predecessors technologies in regards to ActiveX. Microsoft also developed different software platforms based on ActiveX such as: ASP (Active Server Pages), Active Scripting, ADO (ActiveX Data Objects) and ASF (ActiveX Streaming Format) which was renamed to “ Advanced Streaming Format” and later to “Advanced Systems Format”[475]

Lastly, by leveraging Visual C++ we can create an ActiveX using MFC (Microsoft Foundation Class) or ATL (Active Template Library). Due to the fact ActiveX is a legacy technology there are different technologies that can replace it like: HTML5 and JavaScript, modern browser extensions and WebAssembly modules[476]. By the way, ActiveX is even supported as part of “Microsoft Edge”[477]. This is done by leveraging the “Internet Explorer mode”[478].



[473] https://support.microsoft.com/en-us/windows/use-activex-controls-for-internet-explorer-11-25738d05-d357-39b4-eb2f-fdd074bbf347
[474] https://www.howtogeek.com/717016/remembering-activex-controls-the-webs-biggest-mistake/
[475] https://en.wikipedia.org/wiki/ActiveX
[476] https://learn.microsoft.com/en-us/cpp/mfc/activex-controls?view=msvc-170
[477] https://medium.com/@boutnaru/the-windows-process-journey-msedge-exe-microsoft-edge-747e00211a65
[478] https://learn.microsoft.com/en-us/previous-versions/windows/edge-legacy/microsoft-edge-faq

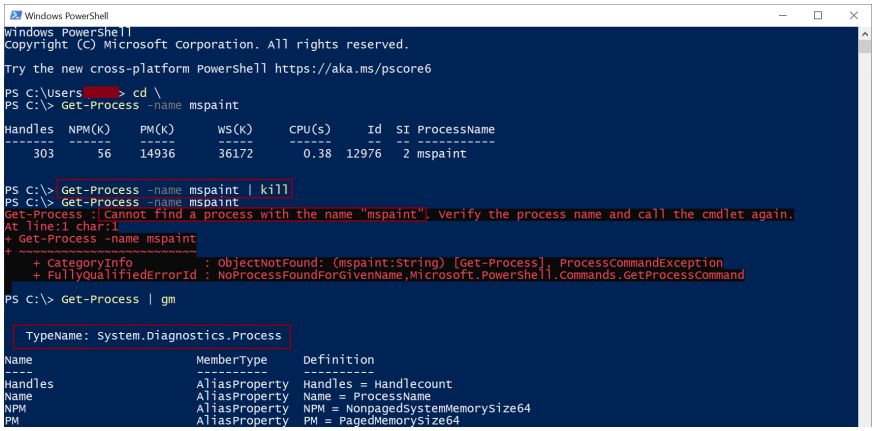
Windows PowerShell

PowerShell is a cross-platform task automation solution which includes: a scripting language, a configuration management framework and a command-line shell. It is important to know that PowerShell can run on Windows, Linux, and macOS[479]. For more information about announcements, features regarding "PowerShell" we can check out Microsoft's on demand video content[480].

Overall, "powershell.exe" (Windows PowerShell) is a PE binary located in "%windir%\system32\WindowsPowerShell\v1.0\powershell.exe". On 64-bit versions we also have a 32-bit version of the binary located at "%windir%\SysWOW64\WindowsPowerShell\v1.0\powershell.exe". By the way, the binary is digitally signed by Microsoft. It is important to know that we can check out the PowerShell source code in its Github repository[481].

Moreover, when developing code we can use the "PowerShell Module Browser" from Microsoft in order to search for modules and cmdlets[482]. A cmdlet is a lightweight command that is used in the PowerShell environment[483]. There is also the "PowerShell Gallery" which is a central repository of PowerShell modules/scripts/DCS resources[484].

Lastly, we can think about "powershell.exe" as a more mature replacement for "cmd.exe"[485]. This is due to the fact we can do anything supported in "cmd.exe" with "powershell.exe" and much more than that. One of the biggest benefits of PowerShell is the fact cmdlets can return as a return value an object and not just a string - as shown in the screenshot below (we call the kill method of the return object).



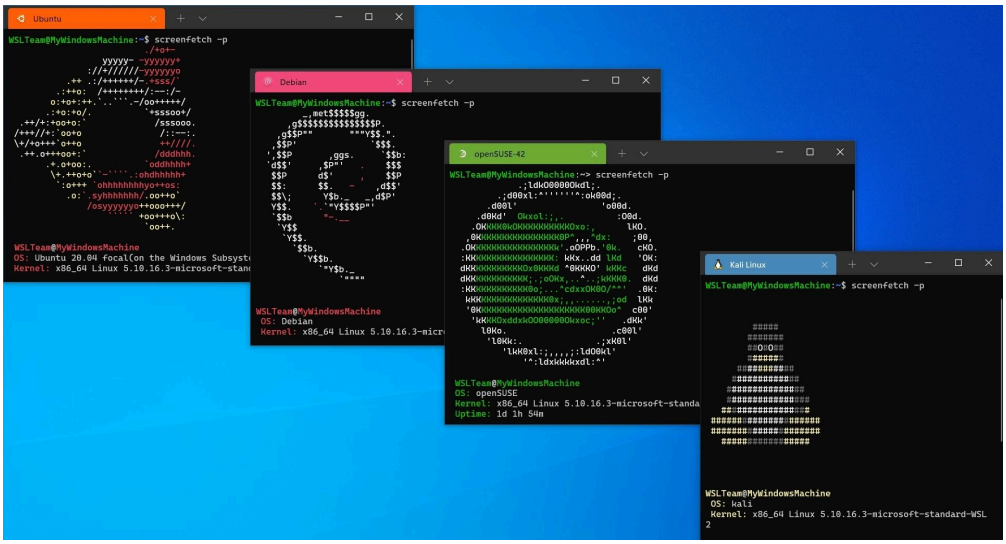
[479] https://learn.microsoft.com/en-us/powershell/scripting/overview
[480] https://learn.microsoft.com/en-us/shows/browse?terms=powershell
[481] https://github.com/PowerShell/PowerShell
[482] https://learn.microsoft.com/en-us/powershell/module/
[483] https://learn.microsoft.com/en-us/powershell/scripting/developer/cmdlet/cmdlet-overview
[484] https://www.powershellgallery.com/
[485] https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b

WSL (Windows Subsystem for Linux)

WSL (Windows Subsystem for Linux) is a feature that is part of the Windows operating system. It can be used for running a Linux environment on top of Windows, this is done without having to run a separate virtual machine by our own and/or dual boot. By leveraging WSL developers can have shameless experience when working with Linux and Windows together[486].

Overall, using WSL we can perform different Linux related tasks such as: installing various Linux distributions (like Ubuntu, Kali and more), storing files in an isolated Linux filesystems of the installed distributions, running Linux command line utilities (like bash, grep, awk and more), running other arbitrary ELF-64 binaries and more[487] - as shown in the screenshot below[488].

Lastly, for reporting issues found on WSL and/or for discussions surrounding WSL we can check WSL's GitHub repo[489]. It is important to know there are two versions of WSL (WSL1 and WSL2) which are based on different architectures - more on those in future writeups. WSL is a replacement of the "Windows Service for Unix" which was deprecated with the release of Windows 8.1[490].



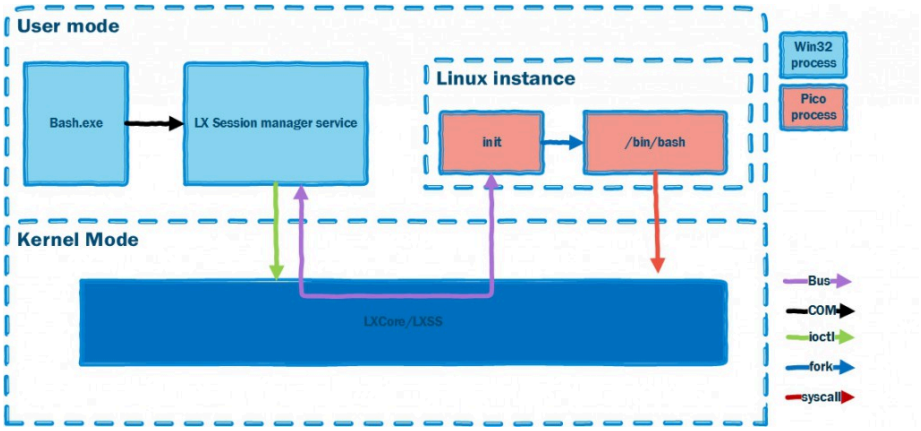
[486] https://learn.microsoft.com/en-us/windows/wsl/
[487] https://learn.microsoft.com/en-us/windows/wsl/about
[488] https://www.deskmodder.de/blog/2023/03/24/wsl-windows-subsystem-for-linux-1-1-6-korrigiert-abstuerze-und-mehr/
[489] https://github.com/microsoft/WSL
[490] https://en.wikipedia.org/wiki/Windows_Services_for_UNIX

WSL1 (Windows Subsystem for Linux version 1)

“WSL 1” is the original version of WSL (Windows Subsystem for Linux) that is used for running a Linux based environment on top of the Windows operating system[491]. It was released in August 2016 in order to act as a compatibility layer for executing ELF (Executable and Linkable Format) files by implementing Linux syscalls[492] as part of the Windows kernel. Not all the Linux binaries are compatible due to the fact not all syscalls were implemented in WSL1[493].

Overall, the architecture of WSL1 is completely different from the architecture of WSL2 (more on that in a future writeup). WSL1 is primarily composed of: a user mode session manager (which manages the Linux instance life cycle), a Pico provider driver (lxss.sys/lxcore.sys) which emulate a Linux kernel by translating Linux syscalls and a Pico process that host the unmodified user mode Linux like “/bin/bash”[494] - as shown in the diagram below.

Lastly, WSL1 does not provide a full Linux kernel and/or a full system call support. By the way, WSL2 is the successor of WSL1 which has a different architecture including an integration with the Linux kernel, increased IO performance and more[495].



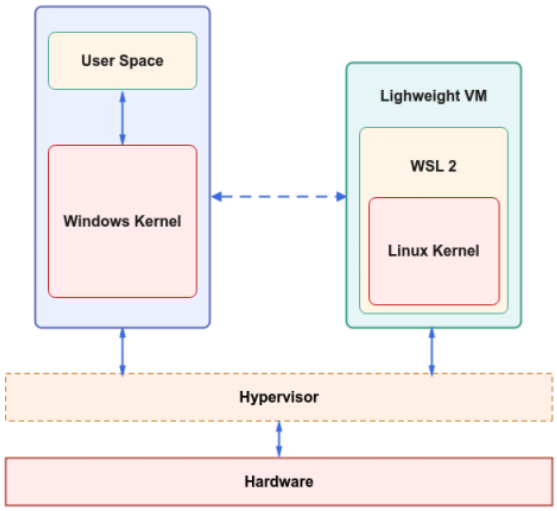
[491] https://medium.com/@boutnaru/the-windows-concept-journey-wsl-windows-subsystem-for-linux-7d629c532110
[492] https://medium.com/@boutnaru/the-linux-concept-journey-syscalls-system-calls-efcd7703e072
[493] https://en.wikipedia.org/wiki/Windows_Subsystem_for_Linux
[494] https://learn.microsoft.com/en-us/archive/blogs/wsl/windows-subsystem-for-linux-overview
[495] https://www.devtopics.com/wsl1-vs-wsl2-a-comparison-and-guide/

WSL2 (Windows Subsystem for Linux version 2)

"WSL 2" (Windows Subsystem for Linux) is the successor version of "WSL 1"[496], which is based on a totally different architecture. WSL 2 leverages virtualization in order to run a Linux kernel as part of a lightweight VM (Virtual Machine). Linux distributions run as an isolated container within the WSL 2 managed VM. The Linux distributions executed under WSL 2 share the same network namespace, device tree, CPU, kernel, memory, swap and the "init" binary. However, they have their own pid namespace, mount namespace, user namespace, cgroup namespace and "init" processes[497].

Overall, WSL 2 was introduced in the middle of 2019. It offers 100% API compatibility with Linux. Thus, we can also execute native Linux GUI applications (X and Wayland). It is important to know that the lightweight utility VM has been optimized to load the Linux kernel into the VM's address space without any boot process. Also, the WSL 2 image files are basically tar files. The architecture of WSL 2 is described in the diagram below[498].

Lastly, the distributions that are supported by WSL 2 without any need for customizations are: Ubuntu, Debian, OpenSUSE, SUSE Enterprise Linux, Kali Linux, Fedora and Pengwin. There are also Linux distributions that can be customized manually to run under WSL 2 like: ArchWSL, AlpineWSL, acme-wsl, miniwsl and more[499].



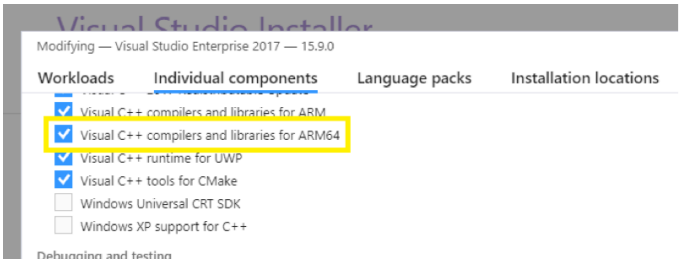
[496] https://medium.com/@boutnaru/the-windows-concept-journey-wsl1-windows-subsystem-for-linux-version-1-7cdb881de548
[497] https://learn.microsoft.com/en-us/windows/wsl/about
[498] https://www.polarsparc.com/xhtml/IntroToWSL2.html
[499] https://www.israelclouds.com/article/wsl-2-0-architecture-and-installation

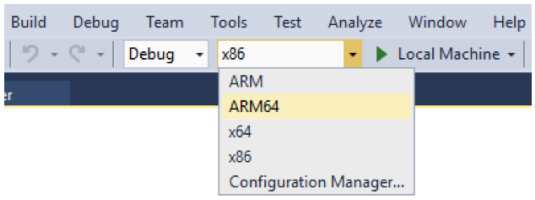
Windows on ARM

Although traditionally Windows runs on systems based on x86/x64 CPUs today it can also run on ARM based computers. We can think about ARM as a SoC (System on Chip) which can include different features such as: CPU, GPU, Wifi and even an NPU (Neural Processor Unit) that accelerates AT workloads (like leveraged by "Copilot PC+). Windows 10 can execute unmodified x86 applications on ARM devices and Windows 11 supports running unmodified x64 applications on ARM devices. Of course for better performance it suggested to execute native applications compiled specifically to ARM[500].

Overall, we can also run workloads on AKS (Azure Kubernetes Service) and on VMs which are ARM based as part of the Azure cloud[501]. There is also official support for ARM development since Windows 10 using "Visual Studio"[502] - as shown in the screenshot below.

Lastly, there are different laptop manufacturers that provide Windows laptops based on ARM processors such as: Microsoft (Surface), Samsung, Asus, and HP. Some of the benefits are: longer battery life and thinner chassis'. There are also tablets and very soon desktops[503].





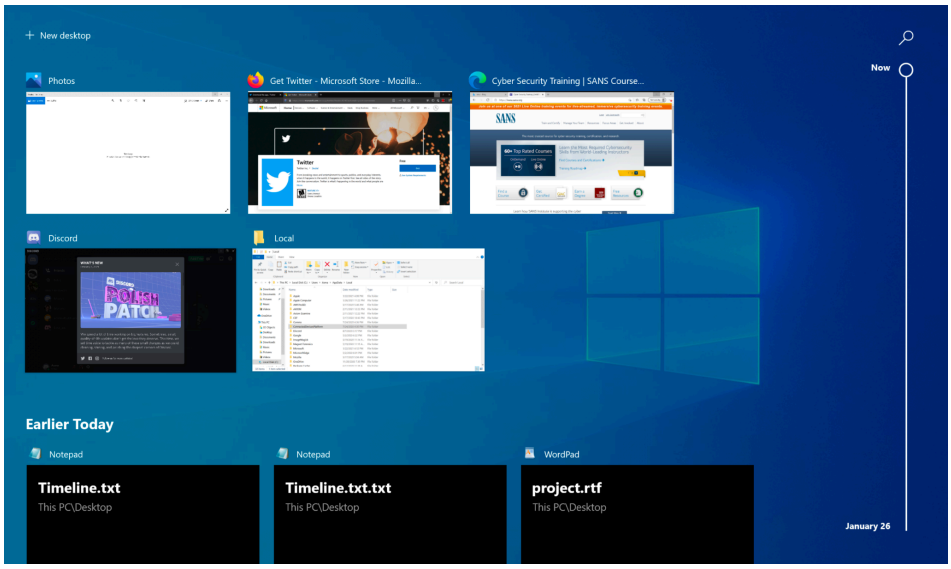
[500] https://learn.microsoft.com/en-us/windows/arm/overview
[501] https://learn.arm.com/learning-paths/servers-and-cloud-computing/aks/cluster_deployment/
[502] https://blogs.windows.com/windowsdeveloper/2018/11/15/official-support-for-windows-10-on-arm-development/
[503] https://www.windowscentral.com/hardware/laptops/best-windows-laptops-with-arm-processor

Windows Timeline

The “Windows Timeline” feature was introduced as part of Windows 10 (version 1803). By using these features a user can checkout current running applications and look back in a timeline on activities done in the past. Examples of such activities are: opened applications/documents/images/videos/websites/etc - as shown in the screenshot below[504].

Overall, we can access the “Windows Timeline” using “WinKey+Tab” or by clicking the “Task View” icon located in the task bar. Also, this feature can be used in order to synchronize activities across different devices[505]. By the way, this feature is also sometimes called “Activity History”.

Lastly, by default the data is stored in “Windows Timeline” for 3-4 days. In case we logon with a Microsoft account the data is stored for up to 30 days[506]. One of the drawbacks is that we can’t limit “Windows Timeline” to stop monitoring a specific application as we can do with “Windows Recall”[507].



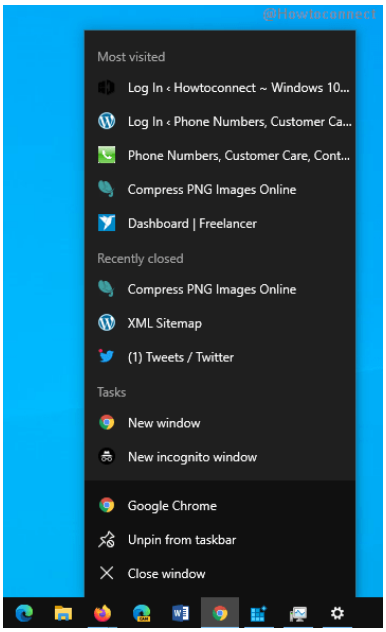
[504] https://forensafe.com/blogs/windowstimeline.html
[505] https://istrosec.com/blog/windows-10-timeline/
[506] https://www.digitalcitizen.life/what-is-timeline-how-use-resume-past-activities/
[507] https://medium.com/@boutnaru/the-windows-forensic-journey-windows-recall-2e31d1844767

Jump List

A “Jump List” is a list of system-provided menus which is shown when the user performs a right-click on an application in the taskbar/start menu. By using jump lists we can access quickly frequent/recently used files (like documents, images and videos) or links in case of browsers[508] - as shown in the screenshot below[509].

Overall, “Jump Lists” are available on the “Start Menu” or using the “Taskbar” while right clicking on an icon of an application[510]. We can enable/disable “Jump Lists” using “Settings->Personalization->Start->Show recently opened items in Jump List on Start or taskbar”.

Lastly, we can also customize “Jump List” by leveraging the “JumpList Class”[511]. Example of such customizations are: adding tasks to the jump list, creating group of items and more[512].



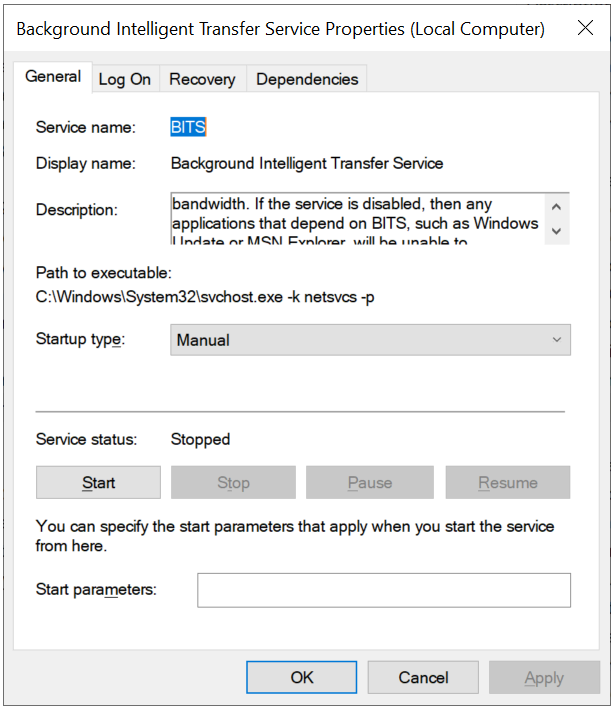
[508] https://learn.microsoft.com/en-us/samples/microsoft/windows-universal-samples/jumplist/
[509] https://www.howto-connect.com/show-jump-lists-on-start-and-taskbar-in-windows-10/
[510] https://www.eiu.edu/busofc/support/pdf/JumpLists.pdf
[511] https://learn.microsoft.com/en-us/uwp/api/windows.ui.startscreen.jumplist?view=winrt-26100&redirectedfrom=MSDN
[512] https://learn.microsoft.com/en-us/samples/microsoft/windows-universal-samples/jumplist/

BITS (Background Intelligent Transfer Service)

BITS (Background intelligent Transfer Service) allows asynchronously transferring files in the foreground\background while controlling the flow of the transfers to preserve the responsiveness of other network applications (like leveraging idle network bandwidth). Also, BITS automatically resumes file transfer in case the transfer is interrupted (think about a disconnect from the network or when restarting the system). BITS uses the Windows "BranchCache" for caching[513].

Overall, BITS has been a part of the Windows operating system since "Windows XP". BITS uses HTTP/SMB protocols for file transfer. BIT is commonly used with "Windows Update"\"Microsoft Update", SCCM (System Center Configuration Manager) and even by "Windows Defender" to fetch signature updates[514]. It is implemented using a "Windows Service"[515] called "Background Intelligent Transfer Service" - as shown in the screenshot below.

Lastly, BITS supports three types of transfer jobs: "Download Job", "Update Job" and "Update-Reply Job". The first, is used for downloading files asynchronously in the foreground or background. The second, is used for uploading files from the client to server. The third, is used for uploading a file and then receiving a reply file from the server[516].



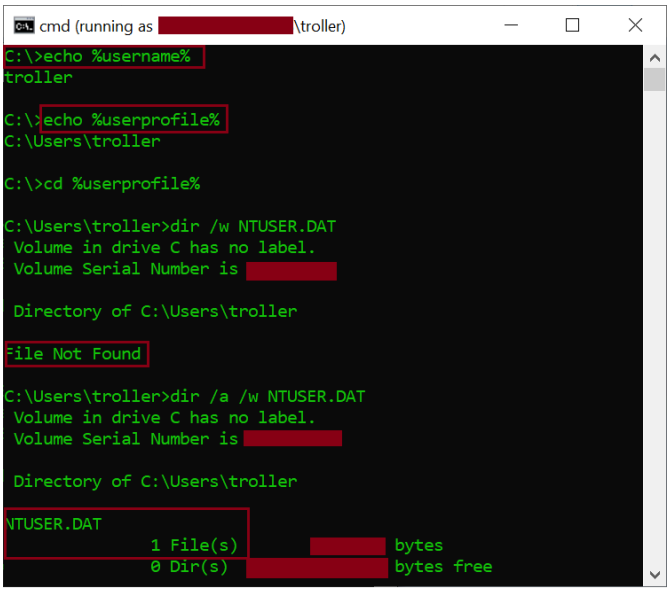
[513] https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/dn282296(v=ws.11)
[514] https://en.wikipedia.org/wiki/Background_Intelligent_Transfer_Service
[515] https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4
[516] https://www.techtarget.com/searchwindowsserver/definition/Microsoft-Windows-Background-Intelligent-Transfer-Service-BITS

User Profile

The first time a user logs on to a Windows device a user profile is created. At every subsequent logon the operating system loads the user's profile. That profile contains configuration of the user environment. Overall, there are the following types of user profiles: "Local User Profile", "Romain User Profile", "Mandatory User Profile" and "Temporary User Profile"[517] - more on each one of them in future writeups.

Moreover, by default the user profiles are created under the "User" folder on the Windows system drive (for example "C:\Users"). Each profile has a separate directory whose name is the %username% (the username of the logged on user). We can use the "%userprofile%" environment variable to access the folder holding the profile of the current user[518].

Lastly, the user profile directory contains profile subfolders (that store different configuration/data on the file system) and a registry hive[519]. This hive is loaded into memory and later mapped as "HKEY_CURRENT_USER". The registry hive is stored in the "NTUSER.DAT" file, which is marked as hidden - as shown in the screenshot below.



[517] https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/bb776892(v=vs.85)
[518] https://www.computerhope.com/issues/ch000109.htm
[519] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9

Local User Account

In general, a “Local User Account” is a Windows account which was created on the local device. This type of account can only logon to a specific device (the local machine) and due to that is used for managing/securing resources on a specific device or as service users[520].

Moreover, the information regarding local user accounts on Windows is stored as part of the SAM[521] (Security Account Manager) which itself is part of the registry. Also, Windows has default local accounts that are created when the operating system is installed. Examples of such users are: Administrator, Guest, DefaultAccount, Local System, Network Service, Local Service[522].

Lastly, we can also list the local users accounts of a specific computer using the “user” argument of the “net.exe” command line utility - as shown in the screenshot below[523]. It is important to note that there are also local groups that can be created.

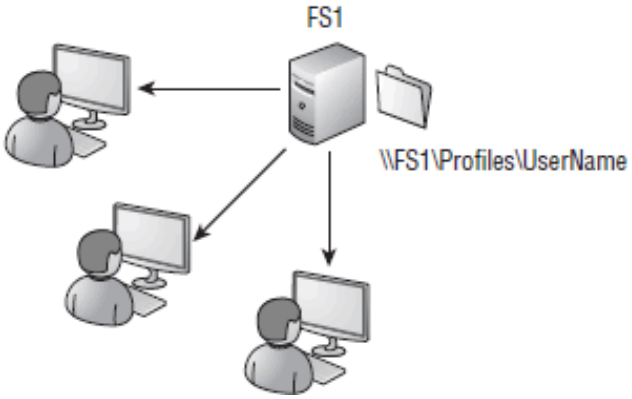
[520] https://learn.microsoft.com/en-us/windows/security/identity-protection/access-control/local-accounts
[521] https://medium.com/@boutnaru/windows-security-sam-security-account-manager-c93ddadf388a
[522] https://learn.microsoft.com/en-us/windows/security/identity-protection/access-control/local-accounts
[523] https://medium.com/@boutnaru/the-windows-process-journey-net-exe-net-command-91e4964f20b8

Roaming User Profile

The goal of a "Roaming User Profile" is to provide users their personal data/settings on every Windows system (or even virtual desktop) that is part of the corporate networks (domain based). In order for it to work roaming profiles are saved on a network share[524] - as shown in the diagram below[525].

Overall, when a user logs on the user's profile is copied from a network share to the local device. When the user performs a logoff[526], the updated profile is uploaded to the relevant network share. This is done to ensure the next time the user logs on the current/correct/updated profile is used. Thus, what we need to do is to create a network share and modify the user account in the domain by setting the path to the user profile to something like that "\\ServerName\ShareName\%UserName%"[527].

Lastly, in order to avoid the case in which roaming profiles take too much space (which can lead to long logon\logoff times) we can use "Folder Redirections". By using folder redirections we can set directories (like downloads) to the local computer which reduces the size of the profile[528].



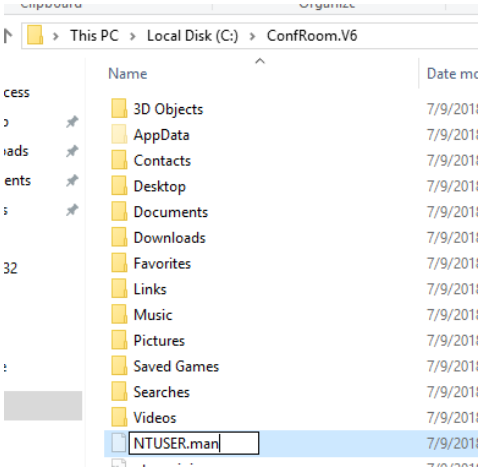
[524] https://www.techtarget.com/searchvirtualdesktop/answer/How-does-a-roaming-user-profile-work
[525] https://www.sourcedaddy.com/windows-7/roaming-profiles.html
[526] https://medium.com/@boutnaru/the-windows-process-journey-logoff-exe-session-logoff-utility-2d1fb2aa7ddd
[527] https://web.archive.org/web/20180704092603/https://www.sourcedaddy.com/windows-7/roaming-profiles.html
[528] https://learn.microsoft.com/en-us/windows-server/storage/folder-redirection/deploy-roaming-user-profiles

Mandatory User Profile

A “Mandatory User Profile” is a special type of a “Roaming User Profile”[529]. In case of a “Mandatory User Profile” when the user makes configuration changes they are not saved when the user logs off[530].

In order to set a “Roaming User Profile” as a “Mandatory User Profile” we just need to rename the name of the registry hive file “NTUSER.DAT”[531] to NTUSER.MAN. Using the “*.man” extension causes the profile to be read-only[532] - as shown in the screenshot below[533].

Lastly, there are two types of a mandatory user profile: “Super-Mandatory User Profile” and “Normal Mandatory User Profile” - more information about each of those in a future writeup. Thus, we can sum up that a mandatory user profile is stored on a network share on a server and that the user cannot modify[534].



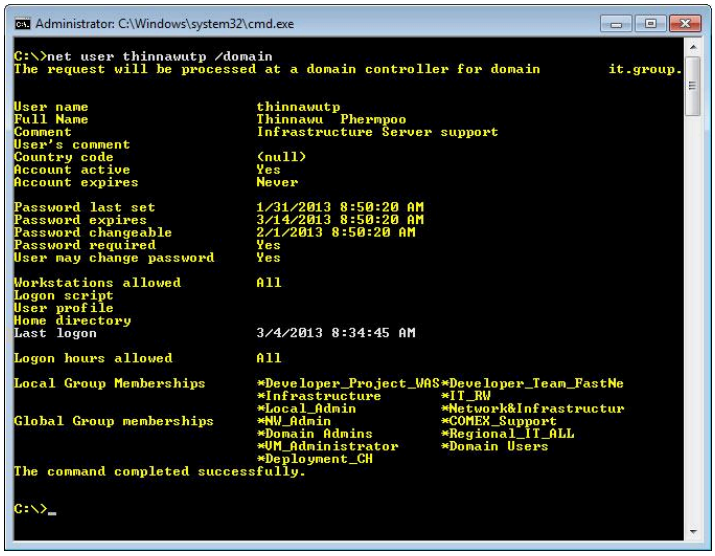
[529] https://medium.com/@boutnaru/the-windows-concept-journey-roaming-user-profile-5228a11785fd
[530] https://learn.microsoft.com/en-us/windows/win32/shell/mandatory-user-profiles
[531] https://medium.com/@boutnaru/the-windows-concept-journey-ntuser-dat-ecdba539b349
[532] https://learn.microsoft.com/en-us/windows/client-management/client-tools/mandatory-user-profile
[533] https://woshub.com/mandatory-user-profiles-windows-10/
[534] http://www.thenetworkencyclopedia.com/entry/mandatory-user-profile/

Domain User Account

By using a domain user account we get the full advantage of the administration and security features provided by "Microsoft Active Directory". Thus, a domain user account provides the ability to log on into systems that are part of a MS domain[535]. As opposed to a local user account which is stored locally in the SAM[536], a domain user account is stored remotely (can be cached locally) in the AD database as part of a "Domain Controller".

Overall, for administering domain user accounts we can use both CLI tools and GUI based tools. Probably the most commonly used GUI tool is "Active Directory and Computers" aka "dsa.msc"[537]. In the case of CLI we can use also the "net.exe"[538] utility which is relevant for local users\group management we just need to add the "/domain" switch to part of the commands - as shown in the screenshot below[539].

Lastly, a domain user account has two name formats: the DN (distinguished name) of the user object and ""<domain>\<username>"[540], where <domain> is the NETBIOS name of the domain or the FQDN (Fully Qualified Domain Name) of the domain. Also, a user in a domain can have a UPN (User Principal Name) which we can think of as a username in an email address format: <username>@<domain>[541].



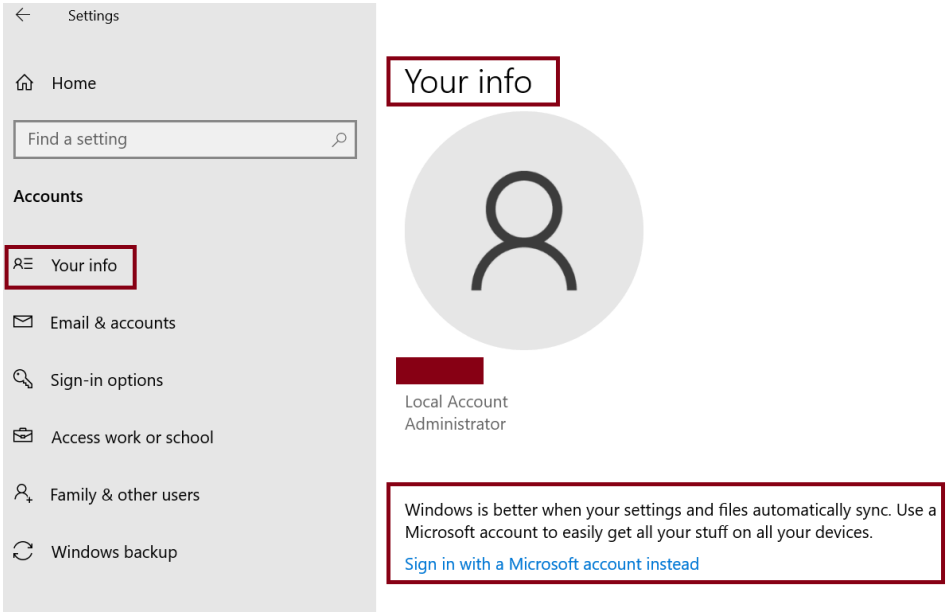
[535] https://medium.com/@boutnaru/the-windows-concept-journey-ad-active-directory-af9e795f86b4
[536] https://medium.com/@boutnaru/the-windows-security-journey-local-user-account-cffe75db39d8
[537] https://woshub.com/install-active-directory-users-computers-aduc-console/
[538] https://medium.com/@boutnaru/the-windows-process-journey-net-exe-net-command-91e4964f20b8
[539] http://serveradmintools.blogspot.com/2013/03/details-and-examples-net-user-domain.html
[540] https://learn.microsoft.com/en-us/windows/win32/ad/domain-user-accounts
[541] https://www.codetwo.com/kb/upn/

MSA (Microsoft Account)

MSA (Microsoft Account) is an SSO (Single Sign On) account that can be used for accessing different Microsoft's services (Bing, Outlook.com, Microsoft Azure and more), devices running the Windows OS (Windows 8 and later\Windows Server 2012 and later) and other software created by Microsoft (Visual Studio, Skype, Windows Movie Maker and more). It was previously named "Microsoft Passport", ".NET Passport" and "Windows Live ID"[542].

Overall, in order to create an MSA we can use an existing email and/or sign up for a Microsoft email. address. The usage of MSA can also be leveraged by organizations by allowing: the download of Microsoft store apps (those which the enterprise wants to distribute), allow for personalized settings across devices, integrated social media services and more[543].

Lastly, is it important to understand that even if we use MSA to log on to the system, we can still leverage local user accounts[544] as a login option. By the way, the same is also true if we use a domain account to log on to our system. If an administrator wants he\she can block the usage for MSA for user authentication using configurations as part of GPO[545].



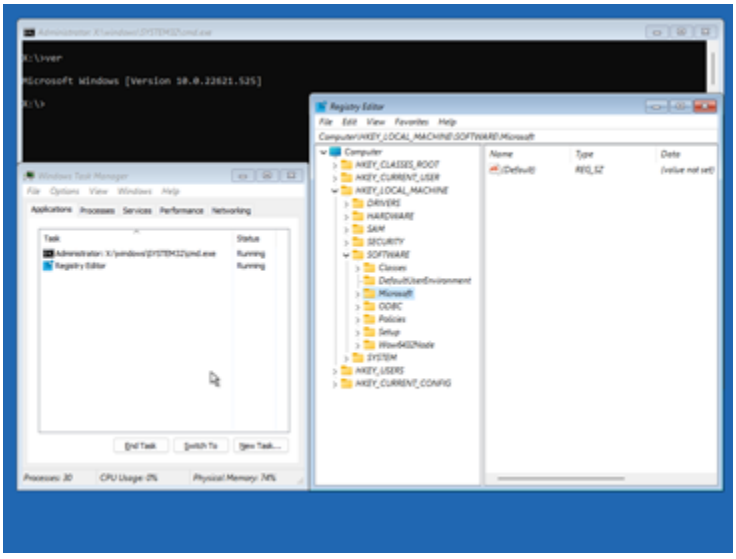
[542] https://en.wikipedia.org/wiki/Microsoft_account
[543] https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/manage/understand-microsoft-accounts
[544] https://medium.com/@boutnaru/the-windows-security-journey-local-user-account-cffe75db39d8
[545] https://medium.com/@boutnaru/the-windows-security-journey-gpo-group-policy-object-f8757933217a

Windows PE (Windows Preinstallation Environment)

Windows PE (Preinstallation Environment), also known as WinPE, is a small operating system. It can be used for installing/deploying/repairing Windows operating systems (desktop/server editions). Using Windows PE we can perform different tasks such as: recovering data from unbootable devices, setup hard drives before installing Windows, altering/modifying the Windows operating system while it is not executing and capturing/applying Windows images[546].

Overall, there are versions of Windows PE built from Windows XP and the latest is built from Windows 11 - as shown in the screenshot below. It was originally created as a preinstall platform for replacing MS-DOS with the Windows operating system. There is also WinRE (Windows Recovery Environment) which is based on WinPE that can be used for diagnosing\recovering in case of serious errors which may prevent Windows from booting successfully[547].

Lastly, we can say that WinPE is a lightweight version of Windows that we can boot using USB/CD/DVD/Hard disk/PXE (think about it as a Windows LiveCD). Thus, it can be used for customizing Windows images for large-scale deployment. Today WinPE is part of Windows ADK (Windows Assessment and Deployment Kit). We can download WinPE directly and freely from Microsoft's official website[548].



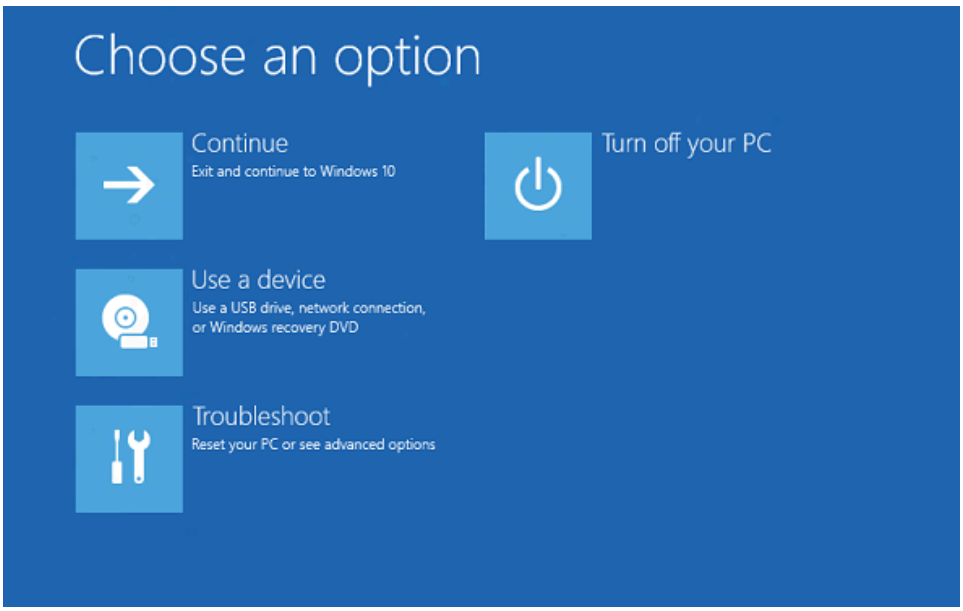
[546] https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/winpe-intro
[547] https://en.wikipedia.org/wiki/Windows_Preinstallation_Environment
[548] https://learn.microsoft.com/en-us/windows-hardware/get-started/adk-install

Windows RE (Recovery Environment)

Windows RE (Recovery Environment), also known as WinRE, is a small operating system. It can be used for repairing common cases that cause Windows not to boot. WinPE[549] is the base layer for WinRE which can be customized with additional languages/drivers/components. A screenshot of WinRE is shown below[550].

Overall, WinRE is the replacement of the "Recovery Console" which was a feature as part of Windows 2000/Windows XP/Windows Server 2003[551]. Thus, WinRE is installed alongside Windows Vista and later, because of that it is included as part of the installation media together with the operating system[552].

Lastly, we can summarize the WinRE's features as follows: automatic repair, system restore for case like unbootable operating system. For accessing WinRE we can use different ways: using a USB recovery media, using the "shutdown /r /o" command, when the computer is turned off we press and hold the WinKey and press the power button and form the Windows logon screen we need to click the power button icon & holds the shift key and click restart[553].



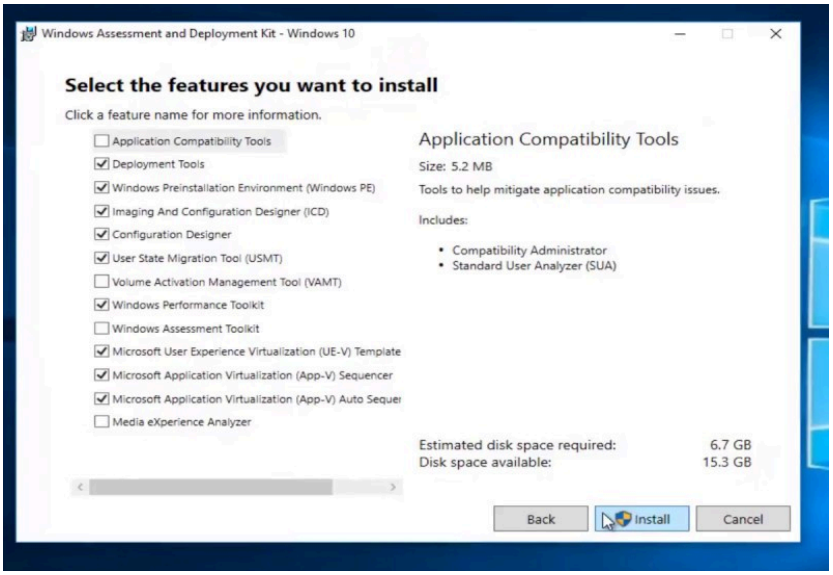
[549] https://medium.com/@boutnaru/the-windows-concept-journey-windows-pe-windows-preinstallation-environment-d77302896112
[550] https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/windows-recovery-environment--windows-re--technical-reference
[551] https://en.wikipedia.org/wiki/Recovery_Console
[552] https://en.wikipedia.org/wiki/Windows_Preinstallation_Environment
[553] https://www.dell.com/support/kbdoc/en-il/000113309/how-to-access-the-windows-recovery-environment-in-windows-10

Windows ADK (Windows Assessment and Deployment Kit)

Windows ADK (Windows Assessment and Deployment Kit) is a collection of tools\technologies created by Microsoft. It is used for deploying Windows OS images to computers and/or virtual drives in VHD format. Windows ADK was previously called "Windows AIK" (Windows Automated Installation Kit) which was first introduced alongside "Windows Vista".

Overall, ADK includes: "Windows Assessment Toolkit" and the "Windows Performance Toolkit" (assess the quality and performance of systems or components), "Windows PE", Sysprep, "Compatibility Administrator", "Standard User Analyzer" and more[554] - as shown in the screenshot below[555].

Lastly, based on the above we can deduct that using Windows ADK we can perform different tasks such as: creating a WinPE deployment engine, cloning system profile from a Windows WIM image, creating Windows PE-based network boot CD/DVD, creation of unattended setup system profile and more[556].



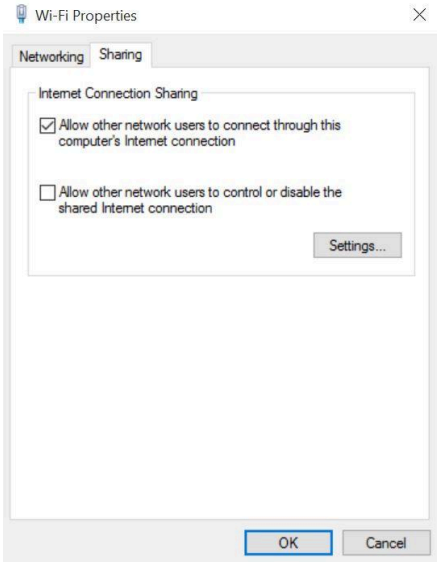
[554] https://learn.microsoft.com/en-us/windows-hardware/get-started/adk-install
[555] https://www.youtube.com/watch?v=XWwgNPtWCNM
[556] https://www.ibm.com/docs/en/tpmfi/7.1.1.17?topic=engine-windows-adk

ICS (Internet Connection Sharing)

Internet Connection Sharing (ICS) is a feature of the Windows operating system that enables a device with Internet access to act as host\access point to other devices[557]. Thus, we can share our internet connection with other computers on our LAN[558]. We can enable\disable ICS using the "Control Panel"[559].

Overall, The specific applet (as part of the control panel) which is used for configuring ICS is "Network Connections" (ncpa.cpl). In order to get to the ICS configuration we can perform the following tasks: "Network Connections"->right click on a LAN/Wi-Fi device->"Properties"->click on the "Sharing" tab->toggle the "Allow other network users to connect through this computer's Internet connection" checkbox - as shown below[560]. We could also use "PowerShell" and\or "netsh.exe" for configuring ICS.

Lastly, ICS provides both DHCP (Dynamic Configuration Host Protocol) and NAT (Network Address Translation) services for the LAN computers. It can also share dial up (PSTN\ISDN\ADSL) , PPoE and VPN connections. Since Windows XP ICS is integrated with UPnP thus remotely discovered/controlled[561]. The relevant settings for ICS are stored in the registry[562] in the following location: "HKLM\SOFTWARE\Policies\Microsoft\Windows\Network Connections[563].



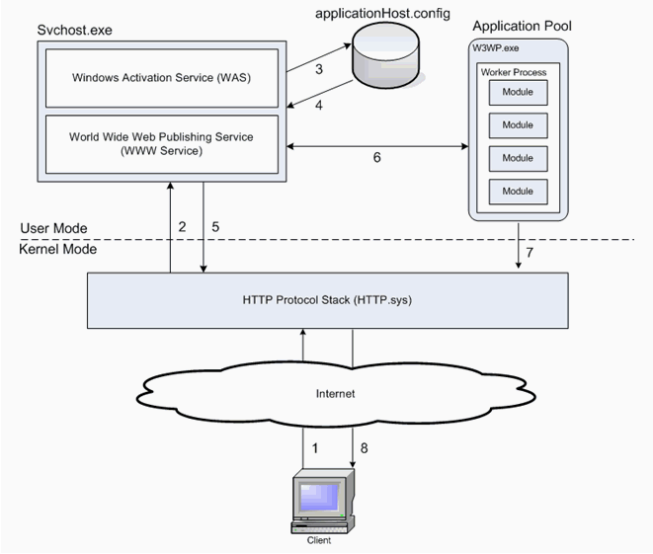
[557] https://support.ringcentral.com/article-v2/Enable-Internet-Connection-Sharing-ICS.html?brand=RC_US&product=RingEX&language=en_US
[558] https://medium.com/@boutnaru/the-nlan-local-area-network-18b3bf4b51d8
[559] https://medium.com/@boutnaru/the-windows-process-journey-control-exe-windows-control-panel-e952c95e2647
[560] http://blog.faister.com/2015/10/15/windows-10-iot-core-raspbian-on-raspberry-pi-2-using-windows-10s-internet-connection-sharing-ics/
[561] https://en.wikipedia.org/wiki/Internet_Connection_Sharing
[562] https://medium.com/@boutnaru/the-windows-concept-journey-registry-0767e79387a9
[563] https://learn.microsoft.com/en-us/windows/privacy/manage-connections-from-windows-operating-system-components-to-microsoft-services

Microsoft IIS (Microsoft Internet Information Services)

Microsoft IIS (Microsoft Internet Information Services) is a flexible Web server supported by the Windows operating system[564]. IIS supports different protocols such as: HTTP, HTTP/2, HTTP/3, HTTPS, FTP, FTPS, SMTP and NNTP[565]. We can compare IIS to other web services like: Apache and NGINX[566].

Overall, until IIS 6.0 the HTTP listener was based on the "Windows Socket API" (Winsock) implemented in user-mode (for receiving and sending HTTP requests). Since IIS 7 (and later) it is based on the "%windir%\System32\drivers\http.sys" kernel mode driver. "Http.sys" provides caching, request queuing, request pre-processing and security filtering[567].

Lastly, until IIS 6 the functionally (administration\configuration, process management and performance monitoring) was handled by a single Windows service[568] called "World Wide Web Publishing Service". From IIS 7.0 the functionally had been splitted between that service and another one called "Windows Process Activation Service". Both are hosted by "svchost.exe"[569] and later can pass the request to the relevant application pool - as shown in the diagram below[570].



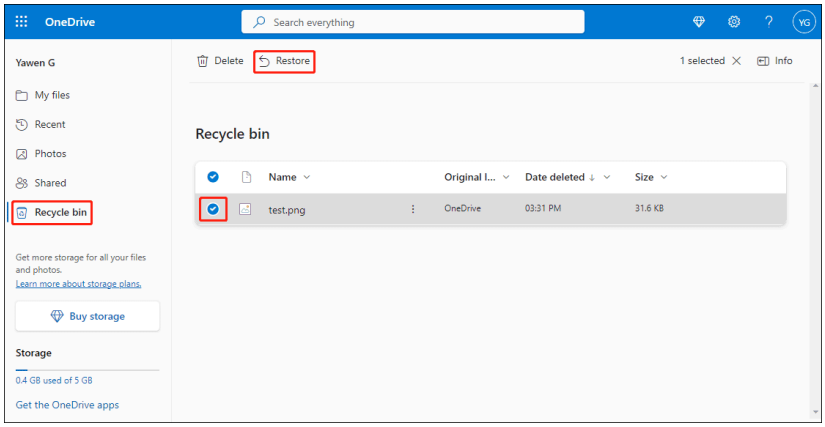
[564] https://www.iis.net/
[565] https://en.wikipedia.org/wiki/Internet_Information_Services
[566] https://vyrazu.com/iis-vs-apache-vs-nginx/
[567] https://learn.microsoft.com/en-us/iis/get-started/introduction-to-iis/introduction-to-iis-architecture
[568] https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4
[569] https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f
[570] https://vanseodesign.com/web-design/iis-web-server/

OneDrive

OneDrive allows keeping files\photos\videos backed up and available across different devices. We can download the OnDrive application for Windows\Android\iOS\macOS based devices[571]. It is a file-hosting service which was released in 2017 (under the name "Windows Live Folders" with the codename of "SkyDrive"). OneDrive can be used also as the storage backend of the web version of "Microsoft 365"[572].

Overall, using OneDrive users can collaborate on work files wherever we go[573]. Every user that has a Microsoft account[574] can access OneDrive. The free tier provides 5GB of storage and can be upgraded. OneDrive can be used by businesses or for personal usage[575].

Lastly, as every service also OnDrive has benefits and disadvantages. Among the benefits are: accessing files/documents any time from different devices and reverting to previous versions (up to 30 days back). Among the disadvantages are: errors in editing (deletion and more) are also synced and can fill up the storage of a device due to synchronization[576]. By the way, we can also recover deleted files by leveraging the "One Drive Recycle Bin" - as shown in the screenshot below. We can do it for 30 days after deletion or 93 if we are signed in using a school/work account[577].



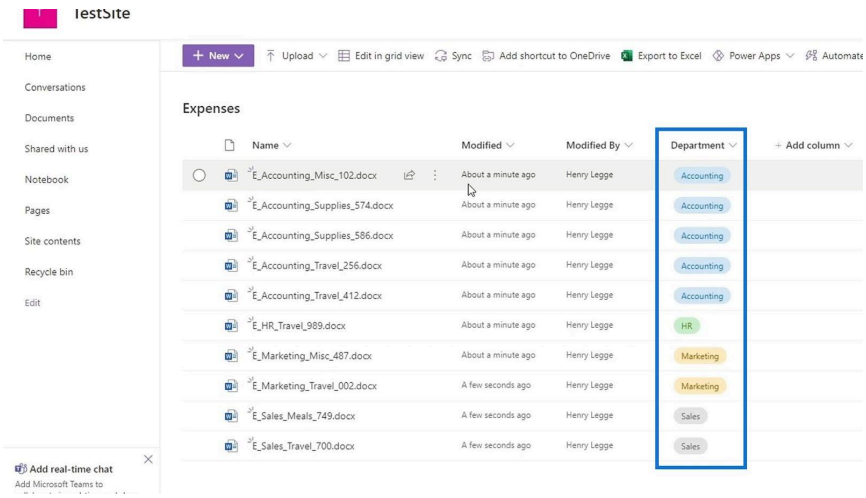
[571] https://www.microsoft.com/en-us/microsoft-365/onedrive/online-cloud-storage
[572] https://en.wikipedia.org/wiki/OneDrive
[573] https://www.microsoft.com/en-us/microsoft-365/onedrive/onedrive-for-business
[574] https://medium.com/@boutnaru/the-windows-security-journey-msa-microsoft-account-ba520c98eff0
[575] https://intranet.ai/articles/microsoft-365/onedrive/
[576] https://www.usn.no/english/about/it-services/onedrive/what-is-onedrive-the-cloud-storage-service
[577] https://www.minitool.com/news/onedrive-recycle-bin.html

SharePoint

SharePoint is a collection of CMS (Content Management System) and knowledge management tools that are developed by Microsoft. It was released in 2021 (formally known as "Windows SharePoint Server" and "Microsoft Office SharePoint Server". SharePoint can be used as an on-premises software and/or as part of "Microsoft 365"[578].

Overall, among the features included by SharePoint we can find: document management (such as indexing, centralized document repository and enterprise search), media asset management (like media library, metadata extraction and built-in media players), record management (for example retention\deletion policy and eDiscovery), web content management (like templeting, page layouts and web content editor), security (such as access control, encryption, 2FA, versioning and audit trail) and more[579].

Lastly, SharePoint is based on a client-server architecture. Users interact with SharePoint using a web browser and/or a mobile application[580]. OneDrive[581] like SharePoint can be used for file sharing\storage - as shown in the screenshot below[582]. OneDrive is basically a simple document library as part of SharePoint. SharePoint leverages OneDrive to store files with a SharePoint site. However, we can use OneDrive without SharePoint[583].



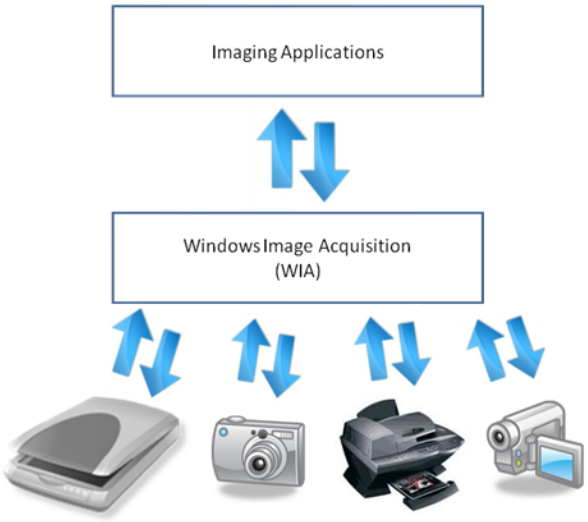
[578] https://en.wikipedia.org/wiki/SharePoint
[579] https://www.scnsoft.com/microsoft/sharepoint/content-management
[580] https://netcomp.com.au/blog/what-is-sharepoint-for-business-and-how-it-works/
[581] https://medium.com/@boutnaru/the-windows-concept-journey-onedrive-c0ba31de7ead
[582] https://quantinsightsnetwork.com/adding-document-library-in-sharepoint/
[583] https://www.nigelfrank.com/insights/everything-you-ever-wanted-to-know-about-microsoft-sharepoint/

WIA (Windows Image Acquisition)

WIA (Windows Image Acquisition) is part of Windows since "Windows ME"\"Windows XP" (WIA 1.0) and used as the still image acquisition platform. Thus, it allows graphics and imaging applications to talk with hardware components (such as scanners and cameras) - as shown in the diagram below[584].

Overall, as part of "Windows Vista" a new version of "Windows Image Acquisition" (WIA 2.0) had been introduced. It added support for "push scanning" which allowed initiating scans and adjusting scanning parameters from the scanner control panel. Also, "multi-image scanning" was added to allow scanning several images at once and saving them directly as separate files. While the video content support had been removed from WIA[585].

Lastly, examples of applications leveraging WIA are: "Photoshop", "Paint.NET", "InfraView", and "ACDSee"[586]. WIA is exposed by the Windows operating system using a Windows service[587] called "stisvc" (HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\stisvc), it is dependent on the "RPC" (Remote Procedure Call) service. The service is implemented as part of "%windir%\system32\wiaservc.dll" and loaded\hosted by "svchost.exe"[588]. In older versions of Windows the service was executed with the permissions of the "Local System" user and was changed (due to security) to the "Local Service"[589].



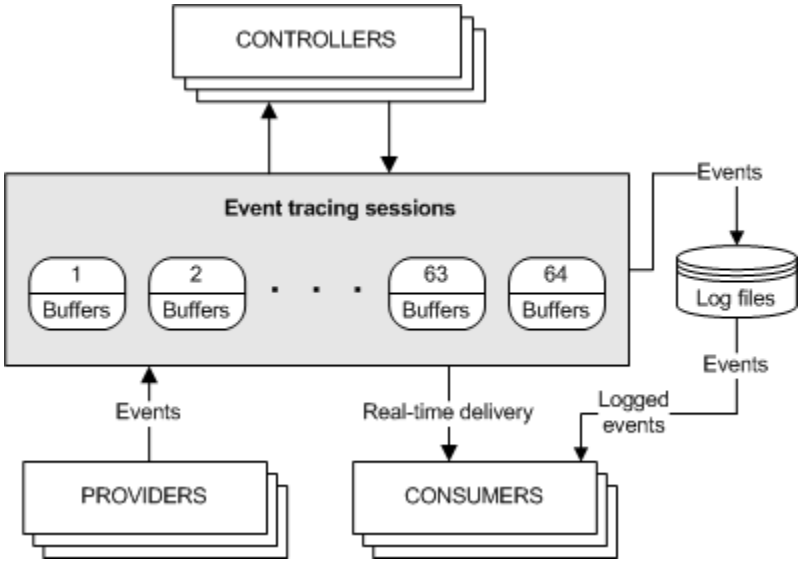
[584] https://learn.microsoft.com/en-us/windows/win32/wia/-wia-startpage
[585] https://www.anvir.com/windows-image-acquisition.htm
[586] https://en.wikipedia.org/wiki/Windows_Image_Acquisition
[587] https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4
[588] https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f
[589] https://medium.com/@boutnaru/the-windows-security-journey-local-service-nt-authority-local-service-b1a624472931

ETW (Event Tracing for Windows)

ETW (Event Tracing for Windows) is a built-in logging mechanism (user-mode and kernel mode) as part of the Windows operating system. It can be used for different use-cases such as: debugging, troubleshooting and even security. By using ETW we can tap different events generated by the Windows operating system[590].

Overall, we can enable/disable event tracing dynamically without the need of restarting the system/applications. The API for using event tracing is composed by: controllers (starting/stopping tracing sessions and enabling providers) providers of events and consumers - as shown in the diagram below (more on that in future writeups). By the way, the total event size (including the ETW header) is 64K[591].

Lastly, there are a number of properties of ETW we should get familiar with: ETW is relevant only for a specific device (no for cross machine), in order to create an ETW trace session administrator privileges (or above) are needed and data information flows asynchronously[592].



[590] https://www.ired.team/miscellaneous-reversing-forensics/windows-kernel-internals/etw-event-tracing-for-windows-101
[591] https://learn.microsoft.com/en-us/windows/win32/etw/about-event-tracing
[592] https://www.preludesecurity.com/blog/event-tracing-for-windows-etw-your-friendly-neighborhood-ipc-mechanism