

The Windows Process Journey

Version 10
May-2024

By Dr. Shlomi Boutnaru



Created using [Craiyon, AI Image Generator](#)

Table of Contents

Table of Contents.....	2
Introduction.....	5
System Idle Process (PID 0).....	6
smss.exe (Session Manager Subsystem).....	7
csrss.exe (Client Server Runtime Subsystem).....	9
wininit.exe (Windows Start-Up Application).....	11
winlogon.exe (Windows Logon Application).....	12
userinit.exe (Userinit Logon Application).....	12
dwm.exe (Desktop Window Manager).....	14
LogonUI.exe (Windows Logon User Interface Host).....	16
explorer.exe (Windows Explorer).....	17
svchost.exe (Host Process for Windows Services).....	18
ctfmon.exe (CTF Loader).....	20
audiodg.exe (Windows Audio Device Graph Isolation).....	21
rdpclip.exe (RDP Clipboard Monitor).....	22
smartscreen.exe (Windows Defender SmartScreen).....	23
ApplicationFrameHost.exe.....	24
RuntimeBroker.exe.....	25
logoff.exe (Session Logoff Utility).....	26
cscript.exe (Microsoft ® Console Based Script Host).....	27
wscript.exe (Microsoft ® Windows Based Script Host).....	28
utilman.exe (Utility Manager).....	29
osk.exe (Accessibility On-Screen Keyboard).....	30
alg.exe (Application Layer Gateway Service).....	31
DrvInst.exe (Driver Installation Module).....	32
runas.exe (Run As Utility).....	33
cmd.exe (Windows Command Processor).....	34
conhost.exe (Console Window Host).....	35
tasklist.exe (Lists the Current Running Tasks).....	36
rundll32.exe (Windows Host Process).....	37
net.exe (Network Command).....	38
net1.exe (Net Command for the 21st Century).....	39
TabTip.exe (Touch Keyboard and Handwriting Panel).....	40
fontdrvhost.exe (Usermode Font Driver Host).....	41
OpenWith.exe (Pick an App).....	42
mavinject.exe (Microsoft Application Virtualization Injector).....	43
where.exe (Lists location of Files).....	44
NisSrv.exe (Microsoft Network Realtime Inspection Service).....	45

Hostname.exe (Hostname APP).....	46
mmc.exe (Microsoft Management Console).....	47
msg.exe (Message Utility).....	48
Magnify.exe (Microsoft Screen Magnifier).....	49
mstsc.exe (Remote Desktop Connection).....	50
curl.exe (cURL executable).....	51
winver.exe (Version Reporter Applet).....	52
arp.exe (TCP/IP Arp Command).....	53
WFS.exe (Microsoft Windows Fax and Scan).....	54
clip.exe (Copies the Data into Clipboard).....	55
consent.exe (Consent UI for Administrative Applications).....	56
getmac.exe (Displays NIC MAC information).....	57
defrag.exe (Disk Defragmenter Module).....	58
msedge.exe (Microsoft Edge).....	59
tzutil.exe (Windows Time Zone Utility).....	60
expand.exe (LZ Expansion Utility).....	61
WSReset.exe (Windows Store Reset).....	62
SlideToShutDown.exe (Windows Slide To Shutdown).....	63
takeown.exe (Takes Ownership of a File).....	64
dialer.exe (Microsoft Windows Phone Dialer).....	65
bthudtask.exe (Bluetooth Uninstall Device Task).....	66
DisplaySwitch.exe (Windows Display Switch).....	67
SpaceAgent.exe (Storage Spaces Settings).....	68
tar.exe (BSD tar Archive Tool).....	69
timeout.exe (Pauses Command Processing).....	70
doskey.exe (Keyboard History Utility).....	71
fsquirt.exe (Bluetooth File Transfer).....	72
label.exe (Disk Label Utility).....	73
forfiles.exe (Execute a Command on Selected Files).....	74
eudcedit.exe (Private Character Editor).....	75
wmplayer.exe (Windows Media Player).....	76
dvdplay.exe (DVD Play Placeholder Application).....	77
comp.exe (File Compare Utility).....	78
find.exe (Find String (grep) Utility).....	79
mspaint.exe (Paint).....	80
services.exe (Service Control Manager).....	81
sc.exe (Service Control Manager Configuration Tool).....	82
phoneactivate.exe (Phone Activation UI).....	83
choice.exe (Offers the User a Choice).....	84
qprocess.exe (Query Process Utility).....	85
rasdial.exe (Remote Access Command Line Dial UI).....	86

waitfor.exe (Wait/Send a Signal Over a Network).....	87
tsdiscon.exe (Session Disconnection Utility).....	88
RunLegacyCPL Elevated.exe (Running Legacy Control Panel Applet in Elevated Mode)..	89
dism.exe (Deployment Image Servicing and Management Tool).....	90
chkdsk.exe (Check Disk Utility).....	91
UserAccountControlSettings.exe (Configuring UAC Settings).....	92
DeviceCensus.exe (Device Information).....	93
MpCmdRun.exe (Microsoft Malware Protection Command Line Utility).....	94
MpDefenderCoreService.exe (Antimalware Core Service).....	95
MsSense.exe (Windows Defender Advanced Threat Protection Service Executable).....	96
lsass.exe (Local Security Authority Process).....	97
Taskmgr.exe (Task Manager).....	98

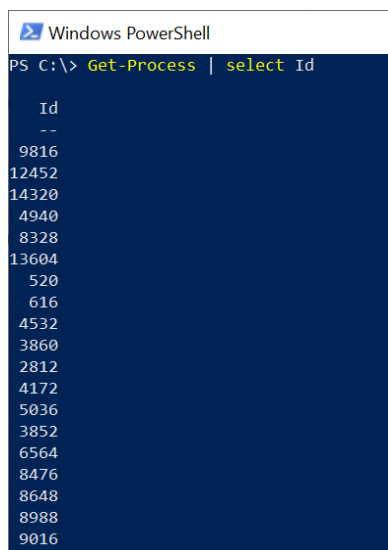
Introduction

Before speaking about a specific process I wanted to talk about an attribute related to all processes on Windows which is not so well known among all administrators/users/programmers etc.

I encourage you before reading the next lines to open any process listing app/program that you like in Windows (tasklist, task manager, process explorer or anything else) and go over PID numbers of all the processes - What can you learn from those numbers?

You probably saw that all of them are even numbers, what is more interesting is that if you divide them by two you will still get an even number - thus all the PIDs are divisible by 4!!!! BTW, the same is true for TIDs (Thread IDs) under Windows. A screenshot from

The reason for that is due to code reuse in the Windows kernel. The PIDs/TIDs are allocated by the same code which allocates kernel handles. Thus, since kernel handles are divisible by 4 so are PIDs/TIDs. We can also use the following powershell command to list only the PIDs: "Get-Process | select ID" - as shown in the screenshot below.



```
Windows PowerShell
PS C:\> Get-Process | select Id

Id
--
9816
12452
14320
4940
8328
13604
520
616
4532
3860
2812
4172
5036
3852
6564
8476
8648
8988
9016
```

But why are the handles divisible by 4? Because the two bottom bits can be ignored by Windows and could be used for tagging. You can verify it by going over the comments in ntdef.h -<https://github.com/tpn/winsdk-10/blob/master/Include/10.0.10240.0/shared/ntdef.h#L846>. Think about the pattern for each PID/TID in binary form to fully understand it.

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!!!!!!

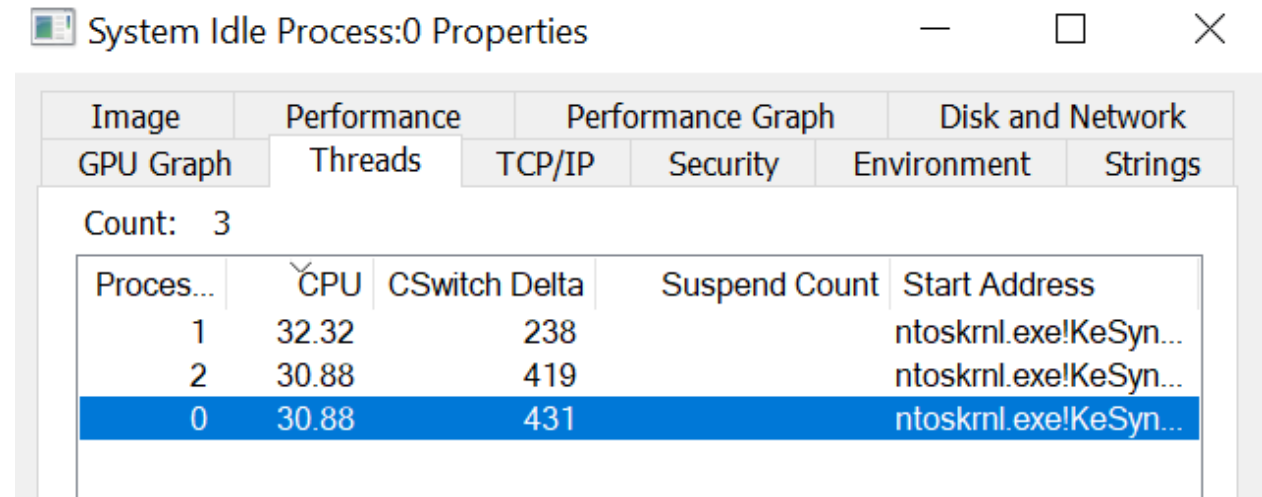
System Idle Process (PID 0)

The goal of this process is to give the CPU something to execute in case there is nothing else to do (thus it is called idle ;-). Let's think about the next situation, we have a process using 30% of CPU, in that case PID 0 (System Idle) will consume the remaining 70%. Also, Idle is the first process that the kernel starts.

Moreover, there is a kernel thread of System Idle for each vCPU the OS has identified (check out the screenshot below which shows that. The VM which I have used had 3 vCPUs - also see the first field in the table showing the "Processor").

The reason for having an "Idle Process" is to avoid an edge case in which the scheduler (Windows schedule based on threads) does not have any thread in a "Ready" state to execute next. By the way, there are also other schedulers IO and Memory, which we will talk about in one of the next posts/writeups.

When the kernel threads are executed they can also perform different power saving tricks regarding the CPU. One of them could be halting different components which are not in use until the next interrupt arrives. The kernel threads can also call functions in the HAL (hardware abstraction layer, more on that in the future) in order to perform tasks such as reducing the CPU clock speed. Which optimization is performed is based on the version of Windows, hardware and the firmware installed.



System Idle Process:0 Properties

Image	Performance	Performance Graph	Disk and Network		
GPU Graph	Threads	TCP/IP	Security	Environment	Strings
Count: 3					
Proces...	Y CPU	CSwitch Delta	Suspend Count	Start Address	
1	32.32	238		ntoskrnl.exe!KeSyn...	
2	30.88	419		ntoskrnl.exe!KeSyn...	
0	30.88	431		ntoskrnl.exe!KeSyn...	

smss.exe (Session Manager Subsystem)

“smss.exe” is the first user-mode process, it is executed from the following location: %SystemRoot%\System32\smss.exe. It's part of Windows since Windows NT 3.1 (1993). Thus, it starts as part of the OS startup phase and performs different tasks such as those we are doing to detail next (The order of writing is not the order of execution).

Performing delayed renaming/file deletion changes based on configuration in the Registry - “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\FileRenameOperations” (for now we should know the Registry central repository for Windows configuration, more on this in the future).

Creation of DOS device mapping based on “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\DOS Devices” such as AUX, CON, PIPE and more (a short explanation could be found here - <http://winapi.freetchsecrets.com/win32/WIN32DefineDosDevice.htm>).

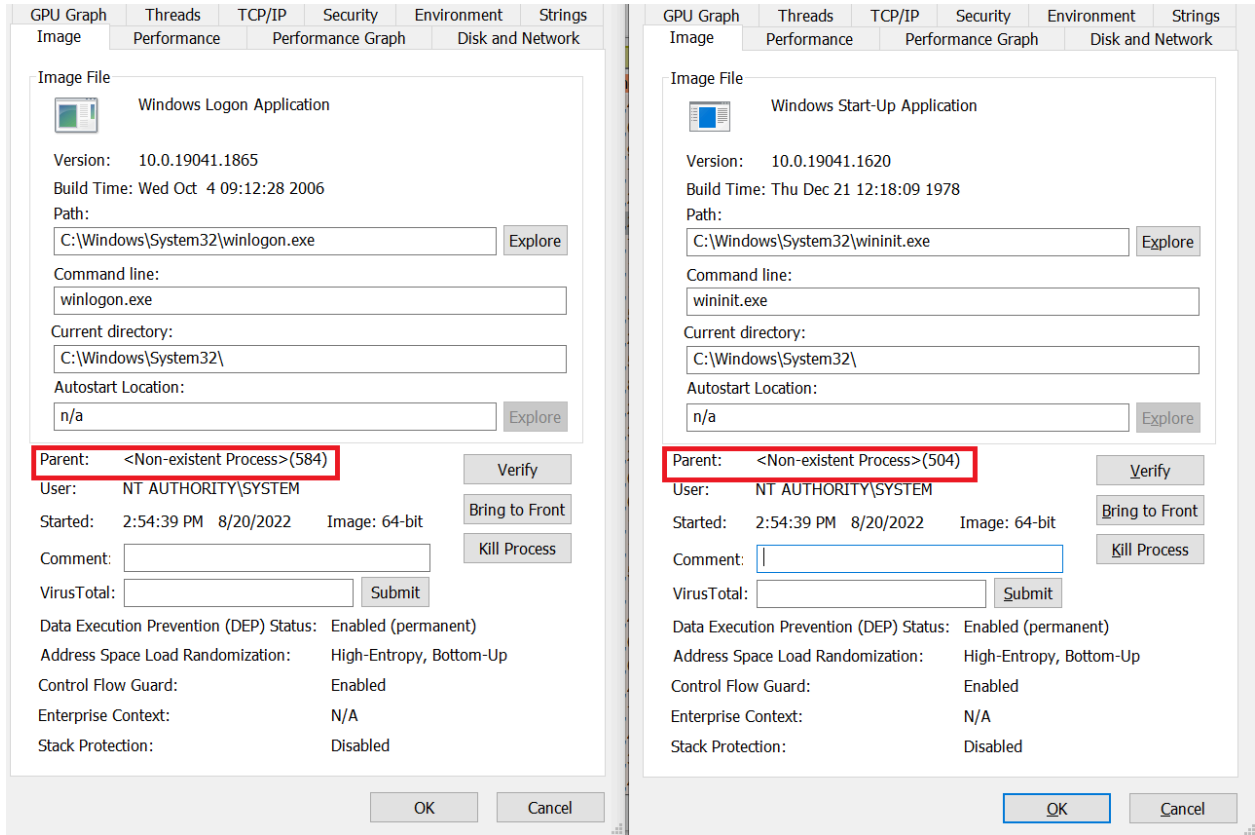
Loading the subsystems which are configured in the Registry - “HKLM\System\CurrentControlSet\Control\Session Manager\SubSystems”. At minimum we have the kernel part of the Win32 Subsystem (aka win32k.sys) and on session 0, which is the session in which Windows' services are executed - smss.exe starts “csrss.exe” and “wininit.exe” (you can also read about them in the following pages).

Also, on session 1, which is the first user session - smss.exe starts “csrss.exe” and “winlogon.exe”. Of course, they could be multiple sessions if more users are logged on (locally or using RDP).

Moreover, both the page files (used for virtual memory) and environment variables (“HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Environment”) are created by “smss.exe”. There are also more actions regarding memory management, KnownDlls, power management and more that are going to be discussed in the future. “smss.exe” also takes part when creating a new RDP session, we will detail this process after taking more in depth about sessions, desktops and windows stations in a future writeup - so stay tuned.

Anyhow, we should expect only one instance of “smss.exe” running without any children processes on session 0, with PPID 4 (“System Process”). This “smss.exe” is called the master, it is responsible for creating at minimum 2 instances of itself for session 0 and 1 (in order to do the work we detailed above). The other instances of “smss.exe” (the non-master) will terminate after finishing the session initialization phase of a new session. On the screenshot below we can see

“wininit.exe” from session 0 and “winlogon.exe” from session 1 both of them having a non-existent parent.



csrss.exe (Client Server Runtime Subsystem)

The goal of “csrss.exe” (Client Server Runtime Subsystem) is to be the user-mode part of the Win32 subsystem (which is responsible for providing the Windows API). “csrss.exe” is included in Windows from Windows NT 3.1. It is located at “%windir%\System32\csrss.exe” (which is most of the time C:\Windows\System32\csrss.exe).

From Windows NT 4.0 most of the Win32 subsystem has been moved to kernel mode - “With this new release, the Window Manager, GDI, and related graphics device drivers have been moved to the Windows NT Executive running in kernel mode”¹. Thus “csrss.exe” manages today GUI shutdowns and windows console (today it is “cmd.exe”).

Overall, we can say that today “csrss.exe” handles things like process/threads, VDM (Visual DOS machine emulation), creating of temp files and more². It is executed by “local system” and there is one instance per user session. Thus, at minimum we will have two (one for session 0 and one for session 1) - as shown in the screenshot below. “csrss.exe” has a handle for each process/thread in the specific session it is part of. Also, for each running process a CSR_PROCESS structure is maintained³, by the way we can leverage this fact for identifying hidden processes (like by using “psxview”⁴ from the volatility framework).

“smss.exe” is the process which starts “csrss.exe” together with “winlogon.exe” (more about it in a future writeup), after finishing “smss.exe” exits. In case you want to read more about “smss.exe”⁵. By the way, from Windows 7 (and later) “csrss.exe” executes “conhost.exe” instead of drawing the console windows by itself (I am going to elaborate about that in the next writeup).

Lastly, “csrss.exe” loads “csrsrv.dll”, “basesrv.dll” and “winsrv.dll” as shown in the screenshot below. If we want to go over some of the source code of “csrss.exe” we can use the ReactOS which is a “A free Windows-compatible Operating System”, which is hosted in github.com. The relevant code of the entire subsystem can be found at <https://github.com/reactos/reactos/tree/master/subsystems/csr>. We can also debug “csrss.exe” using WinDbg, it is important to know that since Windows “csrss.exe” is a protected process so it can be debugged from kernel mode only⁶. A list of all the “csrss.exe” API list can be found here https://j00ru.vexillium.org/csrss_list/api_table.html.

¹[https://learn.microsoft.com/en-us/previous-versions/cc750820\(v=technet.10\)?redirectedfrom=MSDN#XSLTsection124121120120](https://learn.microsoft.com/en-us/previous-versions/cc750820(v=technet.10)?redirectedfrom=MSDN#XSLTsection124121120120)

² <https://j00ru.vexillium.org/2010/07/windows-csrss-write-up-the-basics/>

³ <https://www.geoffchappell.com/studies/windows/win32/csrsrv/api/process/process.htm>

⁴ <https://github.com/volatilityfoundation/volatility/wiki/Command-Reference-Mal#psxview>

⁵ <https://medium.com/@boutnaru/the-windows-process-journey-smss-exe-session-manager-subsystem-bca2cf748d33>

⁶ <https://learn.microsoft.com/en-us/windows-hardware/drivers/debugger/debugging-csrss>

Process Explorer - Sysinternals: www.sysinternals.com (Administrator)

File Options View Process Find Users DLL Help

Process	CPU	Private Bytes	Working Set	PID	Description	Compa...	Protection	Session
csrss.exe	< 0.01	2,092 K	4,616 K	528			PsProtectedSignerWinTcb-Light	0
csrss.exe		1,640 K	3,520 K	612			PsProtectedSignerWinTcb-Light	1
csrss.exe	< 0.01	2,316 K	5,808 K	1276			PsProtectedSignerWinTcb-Light	2

Handles DLLs Threads

Name	Description	Company Name	Path
basesrv.dll	Windows NT BASE API Server DLL	Microsoft Corporation	C:\Windows\System32\basesrv.dll
bcrypt.dll	Windows Cryptographic Primitives Li...	Microsoft Corporation	C:\Windows\System32\bcrypt.dll
bcryptprimitives.dll	Windows Cryptographic Primitives Li...	Microsoft Corporation	C:\Windows\System32\bcryptprimitives.dll
cfgmgr32.dll	Configuration Manager DLL	Microsoft Corporation	C:\Windows\System32\cfgmgr32.dll
combase.dll	Microsoft COM for Windows	Microsoft Corporation	C:\Windows\System32\combase.dll
csrsrv.dll	Client Server Runtime Process	Microsoft Corporation	C:\Windows\System32\csrsrv.dll
csrss.exe	Client Server Runtime Process	Microsoft Corporation	C:\Windows\System32\csrss.exe
csrss.exe.mui	Client Server Runtime Process	Microsoft Corporation	C:\Windows\System32\en-US\csrss.exe.mui
gdi32.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System32\gdi32.dll
gdi32full.dll	GDI Client DLL	Microsoft Corporation	C:\Windows\System32\gdi32full.dll
kernel32.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kernel32.dll
kernelBase.dll	Windows NT BASE API Client DLL	Microsoft Corporation	C:\Windows\System32\kernelBase.dll
locale.nls			C:\Windows\System32\locale.nls
msvcp_win.dll	Microsoft® C Runtime Library	Microsoft Corporation	C:\Windows\System32\msvcp_win.dll
ntdll.dll	NT Layer DLL	Microsoft Corporation	C:\Windows\System32\ntdll.dll
user32.dll	Remote Procedure Call Runtime	Microsoft Corporation	C:\Windows\System32\user32.dll

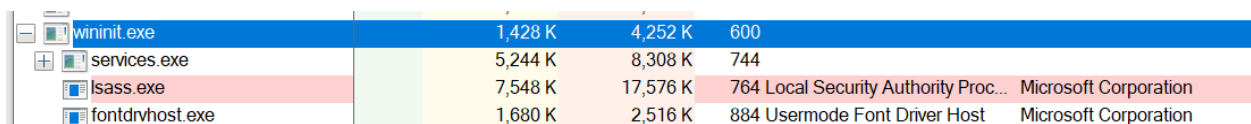
wininit.exe (Windows Start-Up Application)

“wininit.exe” is an executable which is responsible for different initialization steps as described next. The executable is located at “%windir%\System32\wininit.exe” (On 64 bit systems there is only a 64 bit version with no 32 bit version—in contrast to other executables such as cmd.exe). It is started by the first “smss.exe” at session 0 under LocalSystem (S-1-5-18). Overall there should be only one running instance of “wininit.exe”.

Historically, “wininit.exe” was used mainly in order to allow uninstallers to process commands stored in the “WinInit.ini” file. By doing so it allowed programs to take action while the system is booting⁷.

Moreover, “wininit.exe” is responsible for a couple of system initialization steps. Among them are: creating the %windir%\temp folder, initializing the user-mode scheduling infrastructure, creating a window station (Winsta0) and two desktops (Winlogon and Default) for processes to run on in session 0, marking itself critical so that if it exits prematurely and the system is booted in debugging mode (it will break into the debugger) and waiting forever for system shutdown⁸.

Also, “wininit.exe” launches “services.exe” (SCM—Service Control Manager) , “lsass.exe” (Local Security Authority Subsystem) and “fontdrvhost.exe” (Usermode Font Driver Host)—as seen in the screenshot below. If you want more information about service management I suggest reading <https://medium.com/@boutnaru/windows-services-part-1-5d6c2d25b31c> and <https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4>. Regarding the last two (“lsass.exe” and “fontdrvhost.exe”) I am going to write something in the near future.



Process Name	Private Bytes	Working Set	Working Set (K)	Session ID	Company Name
wininit.exe	1,428 K	4,252 K	600		
services.exe	5,244 K	8,308 K	744		
lsass.exe	7,548 K	17,576 K	764	Local Security Authority Proc...	Microsoft Corporation
fontdrvhost.exe	1,680 K	2,516 K	884	Usermode Font Driver Host	Microsoft Corporation

⁷<https://social.technet.microsoft.com/Forums/ie/en-US/df6f5eeb-cbb9-404f-9414-320ea02b4a60/wininitexe-what-is-is-and-why-is-it-constantly-running>

⁸ <https://learn.microsoft.com/en-us/answers/questions/405417/explanation-of-windows-processes-and-dlls.html>

winlogon.exe (Windows Logon Application)

“winlogon.exe” is an executable which is located at “%windir%\System32\winlogon.exe“ (On 64 bit systems there is only a 64-bit version with no 32-bit version like with other executables such as cmd.exe). It is executed under the “NT AUTHORITY\SYSTEM” (S-1-5-18) user. “Winlogon.exe” provides interactive support for interactive logons⁹.

Overall, “winlogon.exe” manages user interactions which are related to the security of the system. Among them are: coordination of the logon flow, handling logout (aka logoff), starting “LogonUI.exe”¹⁰, allowing the alteration of the user’s password and locking/unlocking the server/workstation¹¹. In order to obtain user information for logon “winlogon.exe” uses credentials providers which are loaded by “LogonUI.exe” - more on them in a future writeup. For authenticating the user “winlogon.exe” gets help from “lsass.exe”.

In its initialization phase “winlogon.exe” registers the “CTRL+ALT+DEL” secure attention sequence¹² before any application can do that. Also, “winlogon.exe” creates three desktops within WinSta0: “Winlogon Desktop” (it is the desktop that the user is switched to when SAS is received), “Application Desktop” (this is the desktop created for the logon session of the user) and “ScreenSaver Desktop” (this is the desktop used when a screensaver is running). For more information I suggest reading “Initializing Winlogon”¹³.

Before any logon is performed to the system, the visible desktop is Winlogon’s. Moreover, the number of instances that we expect to have is one for each interactive logon session that is present (as the number of “explorer.exe”) as minimum and in some case another one which is for the next session that can be created - as seen in the screenshot below.

Lastly, I think it is a good idea to go over the reference implementation in ReactOS for “winlogon.exe”¹⁴.

```
C:\>tasklist | findstr explorer.exe
explorer.exe           6568 31C5CE94259D4006      2

C:\>tasklist | findstr winlogon
winlogon.exe          708 Console                1
winlogon.exe          3292 31C5CE94259D4006      2
```

⁹ <https://learn.microsoft.com/en-us/windows/win32/secgloss/w-gly>

¹⁰ <https://medium.com/@boutnaru/the-windows-process-journey-logonui-exe-windows-logon-user-interface-host-4b5b8b6417cb>

¹¹ <https://www.microsoftpressstore.com/articles/article.aspx?p=2228450&seqNum=8>

¹² <https://medium.com/@boutnaru/security-sas-secure-attention-sequence-da8766d859b5>

¹³ <https://learn.microsoft.com/en-us/windows/win32/secauthn/initializing-winlogon>

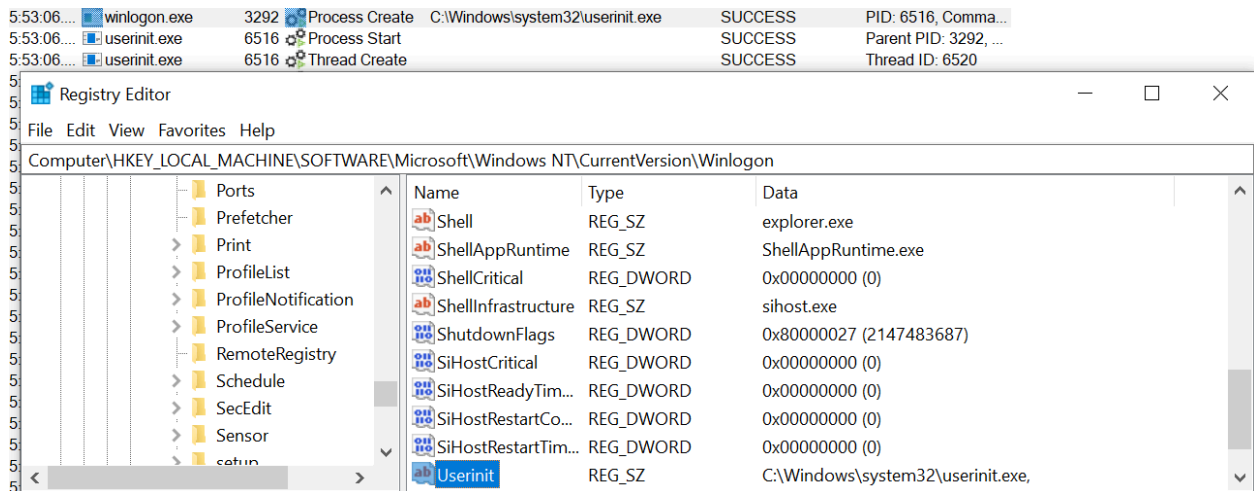
¹⁴ <https://github.com/reactos/reactos/tree/2752e42f0b472f2db873308787a8b474c4738393/base/system/winlogon>

userinit.exe (Userinit Logon Application)

“userinit.exe” is an executable which is located at “%windir%\System32\userinit.exe” (On 64 bit systems there is also a 32 bit version located at “%windir%\SysWOW64\userinit.exe”). It is started by the “winlogon.exe” - as seen in the screenshot below (taken from ProcMon). Also, “userinit.exe” is executed with the permissions of the user which is logging in to the system.

Overall, “userinit.exe” is responsible for loading the user’s profile and executing startup applications while the logon process of the user is being performed. Thus, it will execute logon scripts¹⁵.

“C:\Windows\System32\userinit.exe” is defined by default as the executable for the UserInit phase under the “userinit” key in the registry¹⁶ - as shown in the screenshot below (taken from “regedit.exe”). Moreover, “userinit.exe” runs the shell of the logged on user, which is by default “explorer.exe” as configured in the registry under the “shell” key¹⁷ - as shown in the screenshot below (taken from “regedit.exe”).



I think it is a good idea to go over the reference implementation in ReactOS for “userinit.exe” (<https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aecaf4515603f3f/base/system/userinit>).

¹⁵ <https://www.minitool.com/news/userinit-exe.html>

¹⁶ HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\UserInit

¹⁷ HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell

dwm.exe (Desktop Window Manager)

“dwm.exe” (Desktop Window Manager) is the executable which handles different tasks in the display process of the Windows UI like rendering effects. Among those efforts are: live taskbar thumbnails, Flip3D, transparent windows and more¹⁸. The executable is located at “%windir%\System32\dwm.exe” (On 64 bit systems there is only a 64 bit version with no 32 bit version like with other executables such as cmd.exe).

Thus, we can think about “dwm.exe” as a “compositing windows manager”. A “windows manager” is computer software that controls the placement and appearance of a window as part of a “window system” in a GUI environment¹⁹. So, a “compositing windows manager” is a “window manager” that provides applications with an off-screen buffer for each window. The goal of the manager is to composite all the windows’ buffers into an image representing the screen and commit it to the display memory²⁰.

The desktop composition feature was introduced in Windows Vista. It changed the way applications display pixels on the screen (as it was until Windows XP). When desktop composition is enabled, individual windows no longer draw directly to the screen (or primary display device). Their drawings are redirected to off-screen surfaces in video memory, which are then rendered into a desktop image and presented on the display.

For more information I suggest reading the following links <https://learn.microsoft.com/en-us/windows/win32/dwm/dwm-overview> and https://learn.microsoft.com/en-us/archive/blogs/greg_schechter/under-the-hood-of-the-desktop-window-manager.

Under Windows 10, there is one instance of “dwm.exe” for each session (excluding session 0). The parent process for each “dwm.exe” is “winlogon.exe”. The user which is associated with the security token of each “dwm.exe” has a the pattern of “Window Manager\DWM-{SESSION_ID}” and a SID of pattern “S-1-5-90-0-{SESSION_ID}” as shown in the screenshot below (taken from Process Explorer).

¹⁸ <https://learn.microsoft.com/en-us/windows/win32/dwm/dwm-overview>

¹⁹ https://en.wikipedia.org/wiki/Window_manager

²⁰ https://en.wikipedia.org/wiki/Compositing_window_manager

Process Explorer - Sysinternals: www.sysinternals.com (Administrator)

File Options View Process Find Users Help

Process	CPU	Private Bytes	Working Set	PID	Description	Company Name	Session
winlogon.exe		2,588 K	3,292 K	692	Windows Logon Application	Microsoft Corporation	1
fontdrvhost.exe		1,436 K	1,072 K	864	Usermode Font Driver Host	Microsoft Corporation	1
LogonUI.exe		17,728 K	37,164 K	820	Windows Logon User Interfa...	Microsoft Corporation	1
dwm.exe	< 0.01	27,240 K	28,076 K	956	Desktop Window Manager	Microsoft Corporation	1
csrss.exe	< 0.01	2,160 K	4,408 K	5024			2
winlogon.exe		2,784 K	8,572 K	4328	Windows Logon Application	Microsoft Corporation	2
fontdrvhost.exe		4,408 K	5,136 K	5036	Usermode Font Driver Host	Microsoft Corporation	2
dwm.exe	< 0.01	111,608 K	167,180 K	5308	Desktop Window Manager	Microsoft Corporation	2

dwm.exe:956 Properties

Image	Performance	Performance Graph	Disk and Network	GPU Graph
Threads	TCP/IP	Security	Environment	Strings
User: Window Manager\DWM-1 SID: S-1-5-90-0-1 Session: 1 Logon Session: c91b Virtualized: No Protected: No				
Group ^		Flags		
BUILTIN\Users		Mandatory		
CONSOLE LOGON		Mandatory		
Everyone		Mandatory		
LOCAL		Mandatory		
Mandatory Label\System Mandatory Level		Integrity		
NT AUTHORITY\Authenticated Users		Mandatory		
NT AUTHORITY\INTERACTIVE		Mandatory		
NT AUTHORITY\LOCAL SERVICE		Mandatory		

dwm.exe:5308 Properties

Image	Performance	Performance Graph	Disk and Network
Threads	TCP/IP	Security	Environment
User: Window Manager\DWM-2 SID: S-1-5-90-0-2 Session: 2 Logon Session: 50a4c Virtualized: No Protected: No			
Group ^		Flags	
BUILTIN\Users		Mandatory	
Everyone		Mandatory	
LOCAL		Mandatory	
Mandatory Label\System Mandatory Level		Integrity	
NT AUTHORITY\Authenticated Users		Mandatory	
NT AUTHORITY\INTERACTIVE		Mandatory	
NT AUTHORITY\LOCAL SERVICE		Mandatory	
NT AUTHORITY\This Organization		Mandatory	

LogonUI.exe (Windows Logon User Interface Host)

“LogonUI.exe” (Windows Logon User Interface Host) is responsible for the graphical user interface which asks the user to logon into the system (aka logon screen/lock screen). The executable file is located at “%SystemRoot%\System32\LogonUI.exe” (On 64 bit systems there is only a 64 bit version with no 32 bit version like with other executables such as cmd.exe).

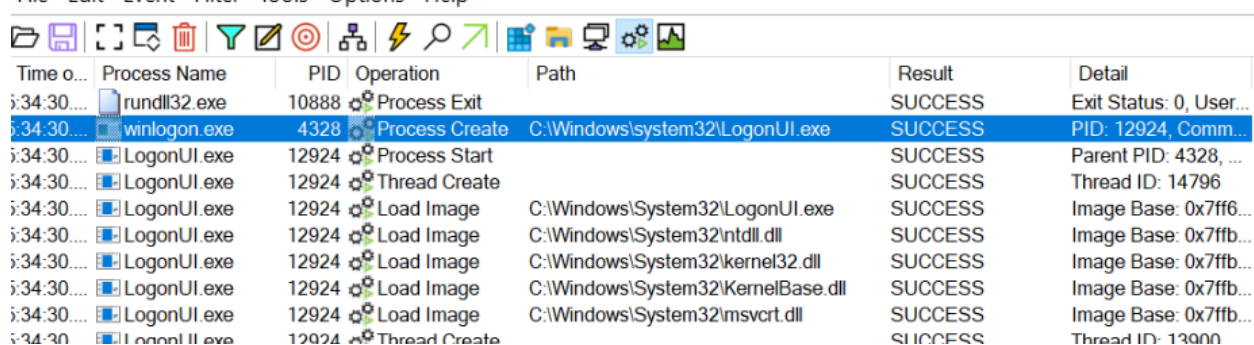
Moreover, “LogonUI.exe” is executed under the Local System user (S-1-5-18) for every session (excluding session 0). “winlogon.exe” is the process which is responsible for running “LogonUI.exe” as we can see in the screenshot below, which was taken from Process Monitor²¹. Also, if you want to see how “LogonUI.exe” GUI looks in different versions of Windows²².

In the perspective of the data flow between “LogonUI.exe” and “winlogon.exe” the basic phases are as follows (after “LogonUI.exe” was launched by “winlogon.exe”). “LogonUI.exe” gets credentials from the user (like username and password) and sends them to “winlogon.exe”. “winlogon.exe” performs the authentication (since Windows Vista it is done using a credential provider, before that it was done by msgina.dll). If the authentication process succeeds, it sends a message back to “LogonUI.exe” to indicate that the user has been authenticated²³. We will get deeper into this flow after talking about “winlogon.exe”, sessions, ALPC (which is the communication line between the processes) and more.

In addition, settings for LogonUI.exe are stored in the registry in the following branch: “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI”. Among those settings we can find the user list that should be shown, the last user that logged-on and the background image. Lastly, if you want to see a reference code for “LogonUI.exe” you can check out ReactOS²⁴.

Process Monitor - Sysinternals: www.sysinternals.com

File Edit Event Filter Tools Options Help



Time o...	Process Name	PID	Operation	Path	Result	Detail
i:34:30...	rundll32.exe	10888	Process Exit		SUCCESS	Exit Status: 0, User...
i:34:30...	winlogon.exe	4328	Process Create	C:\Windows\system32\LogonUI.exe	SUCCESS	PID: 12924, Comm...
i:34:30...	LogonUI.exe	12924	Process Start		SUCCESS	Parent PID: 4328, ...
i:34:30...	LogonUI.exe	12924	Thread Create		SUCCESS	Thread ID: 14796
i:34:30...	LogonUI.exe	12924	Load Image	C:\Windows\System32\LogonUI.exe	SUCCESS	Image Base: 0x7ffb...
i:34:30...	LogonUI.exe	12924	Load Image	C:\Windows\System32\ntdll.dll	SUCCESS	Image Base: 0x7ffb...
i:34:30...	LogonUI.exe	12924	Load Image	C:\Windows\System32\kernel32.dll	SUCCESS	Image Base: 0x7ffb...
i:34:30...	LogonUI.exe	12924	Load Image	C:\Windows\System32\KernelBase.dll	SUCCESS	Image Base: 0x7ffb...
i:34:30...	LogonUI.exe	12924	Load Image	C:\Windows\System32\msvcrt.dll	SUCCESS	Image Base: 0x7ffb...
i:34:30...	LogonUI.exe	12924	Thread Create		SUCCESS	Thread ID: 13000

²¹ <https://learn.microsoft.com/en-us/sysinternals/downloads/procmon>

²² https://media.askvg.com/articles/images3/Windows_Login_Screen.png

²³ <https://learn.microsoft.com/en-us/windows-server/security/windows-authentication/credentials-processes-in-windows-authentication>

²⁴ <https://github.com/reactos/reactos/tree/3647f6a5eb633b52ef4b1db6e43fc2b3fc72969/base/system/logonui>

explorer.exe (Windows Explorer)

“explorer.exe” is an executable which is the “Windows Explorer”. The executable is located at “%windir%\explorer.exe (On 64 bit systems there is also a 32 bit version located in %windir%\SysWOW64\explorer.exe). It is responsible for handling elements of the graphical user interface in Windows (including the taskbar, start menu, and desktop), the “File Explorer” and more. Thus, we can think about it as a graphical shell²⁵.

In case we terminate “explorer.exe” the taskbar will disappear and also the desktop both the shortcuts and the wallpaper itself²⁶. For more understanding about “explorer.exe” I think it is a good idea to go over the reference implementation in ReactOS²⁷.

Every time a user logs in interactively “explorer.exe” is executed under the user which logged on to the system²⁸. The process which starts “explorer.exe” is “userinit.exe” (I will post on it in the near future) - as can be seen in the screenshot below.

11:48:...	userinit.exe	7928	Process Create	C:\WINDOWS\Explorer.EXE	SUCCESS	PID: 11676, Comm...
11:48:...	Explorer.EXE	11676	Process Start		SUCCESS	Parent PID: 7928, ...
11:48:...	Explorer.EXE	11676	Thread Create		SUCCESS	Thread ID: 11692
11:48:...	Explorer.EXE	11676	Load Image	C:\Windows\explorer.exe	SUCCESS	Image Base: 0x7f6...

I also suggest going over the following link <https://ss64.com/nt/explorer.html> to checkout all the arguments that can be passed to “explorer.exe” while launching it. There are also several examples of usage there. By the way, it seems that Microsoft wants to decouple features from “explorer.exe” in order to make Windows 11 faster²⁹.

²⁵ <https://www.pcmag.com/encyclopedia/term/explorerexe>

²⁶ <https://copyprogramming.com/howto/what-happens-if-i-end-the-explorer-exe-process>

²⁷ <https://github.com/reactos/reactos/tree/81db5e1da884f76e6cee66b8cb1c7a2f6ff791eb/base/shell/explorer>

²⁸ <https://learn.microsoft.com/en-us/windows-server/security/windows-authentication/windows-logon-scenarios>

²⁹ <https://www.windowslatest.com/2022/12/22/microsoft-wants-to-make-windows-11-faster-by-decoupling-features-from-explorer-exe/>

svchost.exe (Host Process for Windows Services)

“svchost.exe” is probably the builtin executable which has the most instances (for example 78 on the my testing VM) among all the running processes in Windows. We can split its name to “Svc” and “Host”, that is service host which hits its responsibility (more on that later).

The executable “svchost.exe” is located in %windir%\System32\svchost.exe. In case we are talking about the 64 bit version of Windows, there is also %windir%\SysWOW64\svchost.exe (which is a 32 bit version). Both of the files are signed digitally by Microsoft. It was introduced during Windows 2000, even though there was support for “shared service processes” already in Windows NT 3.1 (more on this in the following paragraphs).

Due to the fact, many of the Windows’ services (you can read on Windows’ Services on <https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4>) are implemented as DLLs (Dynamic Link Libraries) there is a need for an executable to host them. Thus, you can think about “svchost.exe” as the implementation of “shared service process” - A process which hosts/executes/runs multiple services in a single memory address space.

The configuration of services is stored in the registry (“HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services”), for each service which is hosted the name of the DLL is stored under the “Parameter” subkey in a value named “ServiceDll”. For example, in the case of the DHCP client is “HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Dhcp\Parameters\ServiceDll” - as shown in the screenshot below. The ImagePath (which stores the path to the executable to run when starting the service) will be “svchost.exe” with a command line parameter of “-k” and the name of the service groups (like netsvcs, Dcomlaunch, utcsvc, and LocalServiceNoNetwork, LocalSystemNetworkRestricted).

At the end services are splitted into different groups, every group is hosted by one host process which is a single instance of “svchost.exe”. If we want to see which services are hosted on which “svchost.exe” you can use tools like “Process Explorer” and “tasklist” - as you can see in the screenshot below. The configuration of which services are part of what group we can see at “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Svchost” (on my test VM a total of 49 groups are defined).

It is important to know that from Windows 10 (version 1903) on systems with more than 3.5GB or RAM by default there is no grouping. That is, every service will be executed in a single instance of “svchost.exe” for better security and reliability. Of course there are exceptions for that³⁰.

³⁰ <https://learn.microsoft.com/en-us/windows/application-management/svchost-service-refactoring>

Name	Type	Data
(Default)	REG_SZ	(value not set)
ServiceDll	REG_EXPAND_SZ	%SystemRoot%\system32\dhcpcore.dll
ServiceDllUnload...	REG_DWORD	0x00000001 (1)

```

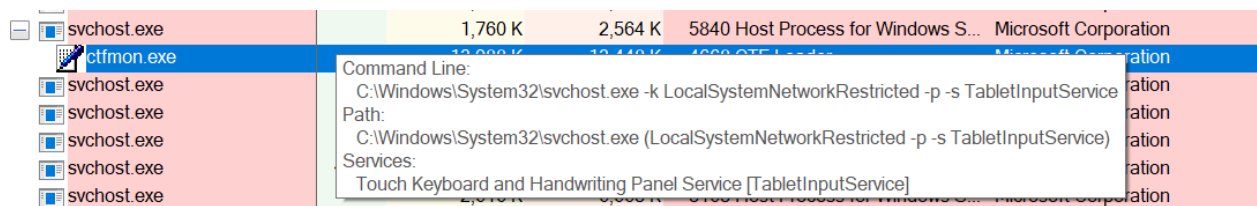
C:\Windows\system32\cmd.exe
services.exe          748 N/A
lsass.exe            768 KeyIso, SamSs, VaultSvc
svchost.exe          880 BrokerInfrastructure, DcomLaunch, PlugPlay,
                    Power, SystemEventsBroker
fontdrvhost.exe      904 N/A
fontdrvhost.exe      912 N/A
svchost.exe          992 RpcEptMapper, RpcSs
svchost.exe          492 LSM
LogonUI.exe          744 N/A
dwm.exe              796 N/A
svchost.exe          1028 TermService
svchost.exe          1104 NcbService
svchost.exe          1112 TimeBrokerSvc
svchost.exe          1188 EventLog
svchost.exe          1224 vmicheartbeat
svchost.exe          1232 vmickvpexchange
svchost.exe          1248 vmicrdv
svchost.exe          1288 vmicshutdown
svchost.exe          1296 nsi
svchost.exe          1356 vmictimesync
svchost.exe          1384 vmicvss
svchost.exe          1496 Dhcp
svchost.exe          1528 ProfSvc
svchost.exe          1556 EventSystem
svchost.exe          1576 SysMain
svchost.exe          1592 Themes
Memory Compression   1720 N/A
VSSVC.exe            1756 VSS
svchost.exe          1764 SENS
    
```

ctfmon.exe (CTF Loader)

“ctfmon.exe” is a user-mode process which is executed from the following location %SystemRoot%\System32\ctfmon.exe. If you are using a 64 bit version of Windows, there is also a 32 bit version of “ctfmon.exe” located at C:\Windows\SysWOW64\ctfmon.exe. By parsing the file information we can see that it is described as a “CTF Loader”. CTF stands for “Collaboration Translation Framework”, it is used by Microsoft Office.

The goal of “ctfmon.exe” is to provide different input capabilities for users such as speech and handwriting recognition. By the way, it will run even if you are not using Microsoft Office.

“Ctfmon.exe” is launched as a child process of the service TabletInputService ("Touch Keyboard and Handwriting Panel Service"), which is hosted by “svchost.exe” - as shown in the screenshot below. Thus, if we want to stop “ctfmon.exe” we can just disable/stop that service. For more information about what is “svchost.exe” you can read the following link <https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f>.

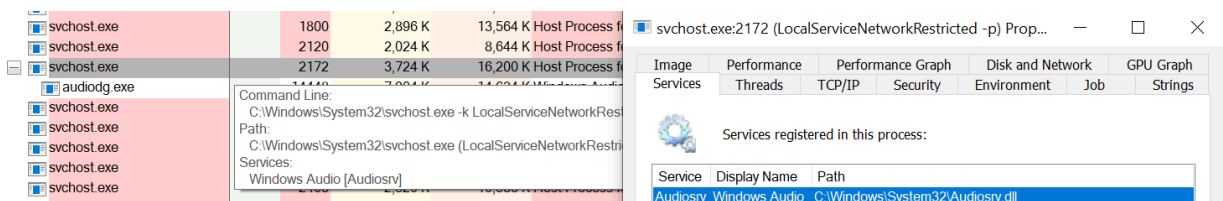


audiodg.exe (Windows Audio Device Graph Isolation)

“audiodg.exe” is an executable which is part of the Windows shared-mode audio engine as described next. The executable is located at “%windir%\System32\audiodg.exe” (On 64 bit systems there is only a 64 bit version with no 32 bit version—in contrast to other executables such as cmd.exe). The process is running under the user “NT AUTHORITY\LOCAL SERVICE”.

In Windows the audio engine runs in user mode. We have the "Windows Audio" service which is implemented in AudioSrv.dll, it is hosted using the “svchost.exe” process. The service launches a helper process “audiodg.exe”³¹. All of that is demonstrated in the screenshot below. It runs in a different login session from the logged on user (isolated) in order to that content and plug-ins cannot be modified³².

Thus, we can say that “audiodg.exe” is being utilized for all audio processing³³. It hosts the audio engine for Windows so all the digital signal processing (DSP) is performed by “audiodg.exe”. Vendors can install their own audio effects which will be processed by “audiodg.exe”³⁴. There should be one instance only of “audiodg.exe” at a specific time.



³¹ <https://learn.microsoft.com/en-us/windows-hardware/drivers/dashboard/audio-measures>

³² <https://answers.microsoft.com/en-us/windows/forum/all/audiodgexe/0c86aef4-81a5-480e-9389-d9652fee1d21>

³³ <https://answers.microsoft.com/en-us/windows/forum/all/windows-10-audiodgexe/af1b70e0-06fe-4952-8205-b6191ccb8882>

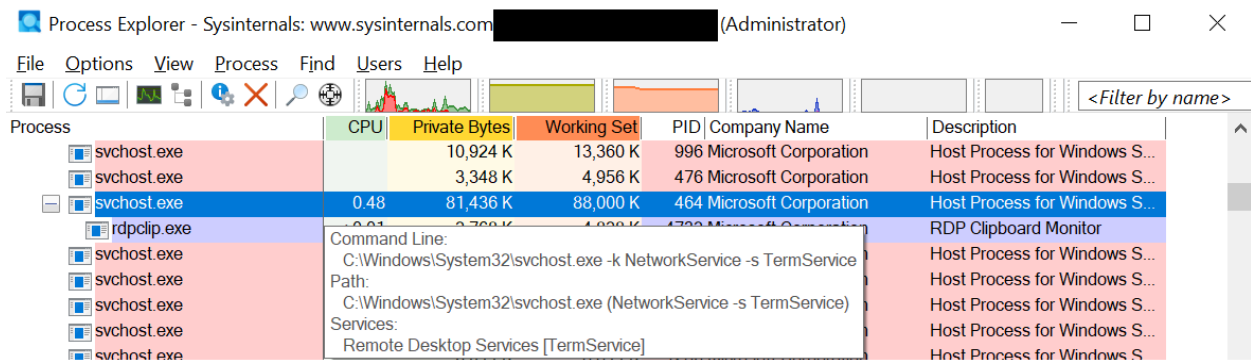
³⁴ <https://answers.microsoft.com/en-us/windows/forum/all/audiodgexe-high-cpu-and-memory/42b3f122-87bf-45cd-8ea7-08abafa9442c>

rdpclip.exe (RDP Clipboard Monitor)

“rdpclip.exe” (RDP Clipboard Monitor) is responsible for managing the shared clipboard between the local computer and the remote desktop which the user is interacting with³⁵. The executable file is located at “%windir%\System32\rdpclip.exe” (On 64 bit systems there is only a 64 bit version with no 32 bit version like with other executables such as cmd.exe).

By enabling the “Remote Desktop” capability³⁶ on Windows it allows remote management of a system using a GUI (graphical user interface) by leveraging the Remote Desktop Protocol (RDP). The default port of the protocol is TCP/3389. For more information about the protocol I suggest reading the following link <https://www.cyberark.com/resources/threat-research-blog/explain-like-i-m-5-remote-desktop-protocol-rdp>.

“rdpclip” is started when a new remote desktop session is created by the service which is called “Remote Desktop Services” - as shown in the screenshot below. Fun fact, the old display name of the service was “Terminal Services” which was changed while the service name is still “TermService”.



Lastly, the description of the service states “it allows users to connect interactively to a remote computer. Remote Desktop and Remote Desktop Session Host Server depend on this service. To prevent remote use of this computer, clear the checkboxes on the Remote tab of the System properties control panel”.

³⁵ <https://www.winosbite.com/rdpclip-exe/>

³⁶ <https://learn.microsoft.com/en-us/windows-server/remote/remote-desktop-services/clients/remote-desktop-allow-access>

smartscreen.exe (Windows Defender SmartScreen)

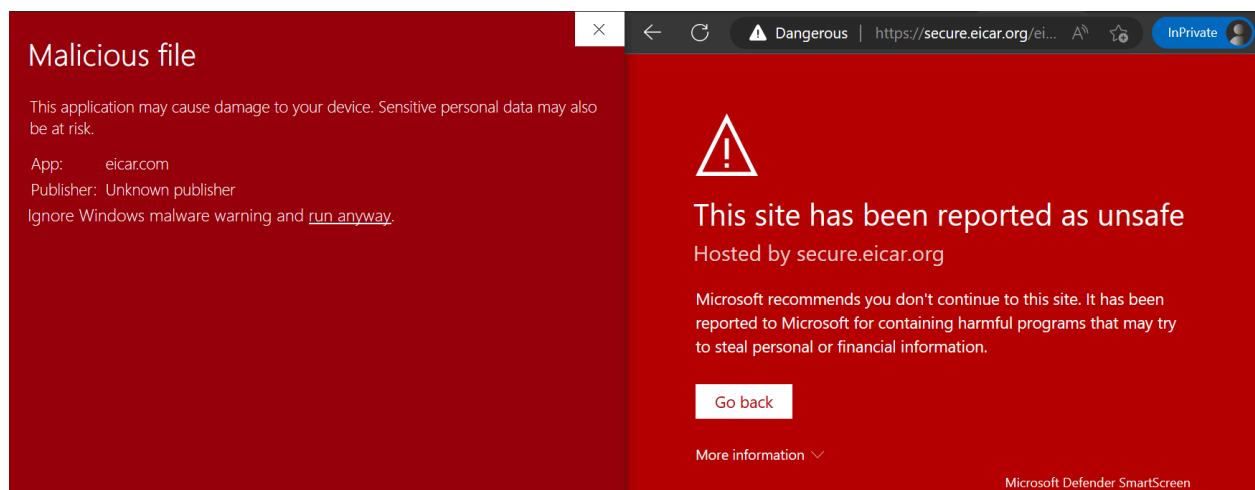
“smartscreen.exe” is an executable which is the “Windows Defender SmartScreen”. The executable is located at “%windir%\System32\smartscreen.exe” (On 64 bit systems there is only a 64 bit version with no 32 bit version—in contrast to other executables such as cmd.exe).

SmartScreen is a cloud-based anti-phishing/anti-malware component which is included in different Microsoft products such as: Windows, Internet Explorer and Microsoft Edge (https://en.wikipedia.org/wiki/Microsoft_SmartScreen).

Microsoft Defender SmartScreen helps with determining whether a site is potentially malicious and by determining if a downloaded application/installer is potentially malicious. We can sum up the benefits of SmartScreen as follows: anti-phishing/anti-malware support, reputation-based URL/application protection, operating system integration, ease of management using group policy/Microsoft Intune and blocking URLs associated with potentially unwanted applications. (<https://learn.microsoft.com/en-us/windows/security/threat-protection/microsoft-defender-smartscreen/microsoft-defender-smartscreen-overview>).

In order to demonstrate the working of SmartScreen I have tried to download (using Edge) - you can see the warning in the left side of the screenshot below. Moreover, after downloading it using a different browser I have executed the EICAR test file - you can see the result in the left side of the screenshot below. By the way, the EICAR (European Institute from Computer Antivirus Research) test file was created to test the response of AV software (https://en.wikipedia.org/wiki/EICAR_test_file).

Lastly, we can enable/disable SmartScreen using the settings window, bot for the OS/browser (<https://www.digitalcitizen.life/how-disable-or-enable-smartscreen-filter-internet-explorer-or-windows-8/>).

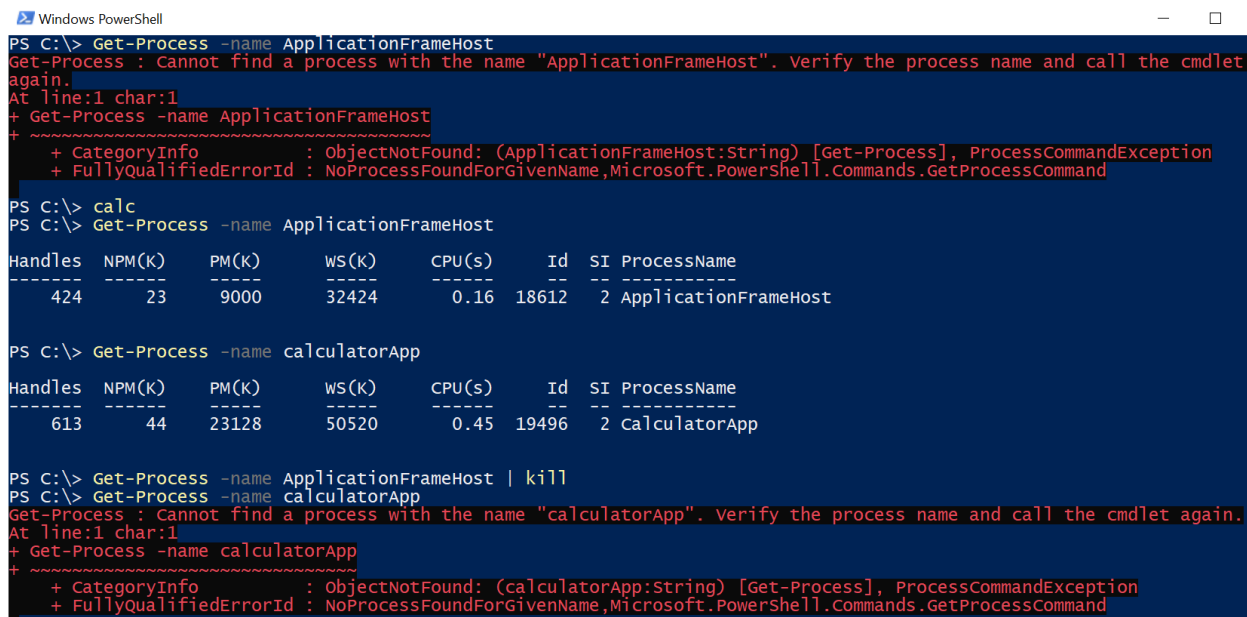


ApplicationFrameHost.exe

The “ApplicationFrameHost.exe” executable is located at the following directory - ”%windir%\system32\ApplicationFrameHost.exe”. On 64-bit systems there is only a 64-bit version with no 32 bit version—in contrast to other executables such as cmd.exe.

Overall, the goal of “ApplicationFrameHost.exe” is to display the frames (windows) of the applications whether we are in desktop/tablet mode³⁷. By the way, if we kill “ApplicationFrameHost.exe” all the UWP applications will be closed also - as we can see in the screenshot below.

There is one instance per session for the “ApplicationFrameHost.exe” in case one or more “Window Store App” which is also known as “Universal Windows Platform App”³⁸ - I will elaborate about them in a separate writeup. An example for a UWP app is the Calculator (“%windir%\system32\calc.exe”). Also, “ApplicationFrameHost.exe” is running with the permissions of the logged on user (that from whom the session was created).



```
Windows PowerShell
PS C:\> Get-Process -name ApplicationFrameHost
Get-Process : Cannot find a process with the name "ApplicationFrameHost". Verify the process name and call the cmdlet again.
At line:1 char:1
+ Get-Process -name ApplicationFrameHost
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (ApplicationFrameHost:String) [Get-Process], ProcessCommandException
+ FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProcessCommand

PS C:\> calc
PS C:\> Get-Process -name ApplicationFrameHost

Handles   NPM(K)    PM(K)      WS(K)      CPU(s)     Id   SI ProcessName
-----
424        23       9000      32424      0.16      18612  2 ApplicationFrameHost

PS C:\> Get-Process -name calculatorApp

Handles   NPM(K)    PM(K)      WS(K)      CPU(s)     Id   SI ProcessName
-----
613        44      23128      50520      0.45      19496  2 calculatorApp

PS C:\> Get-Process -name ApplicationFrameHost | kill
PS C:\> Get-Process -name calculatorApp
Get-Process : Cannot find a process with the name "calculatorApp". Verify the process name and call the cmdlet again.
At line:1 char:1
+ Get-Process -name calculatorApp
+ ~~~~~
+ CategoryInfo          : ObjectNotFound: (calculatorApp:String) [Get-Process], ProcessCommandException
+ FullyQualifiedErrorId : NoProcessFoundForGivenName,Microsoft.PowerShell.Commands.GetProcessCommand
```

³⁷ <https://www.howtogeek.com/325127/what-is-application-frame-host-and-why-is-it-running-on-my-pc/>

³⁸ <https://www.file.net/process/applicationframehost.exe.html>

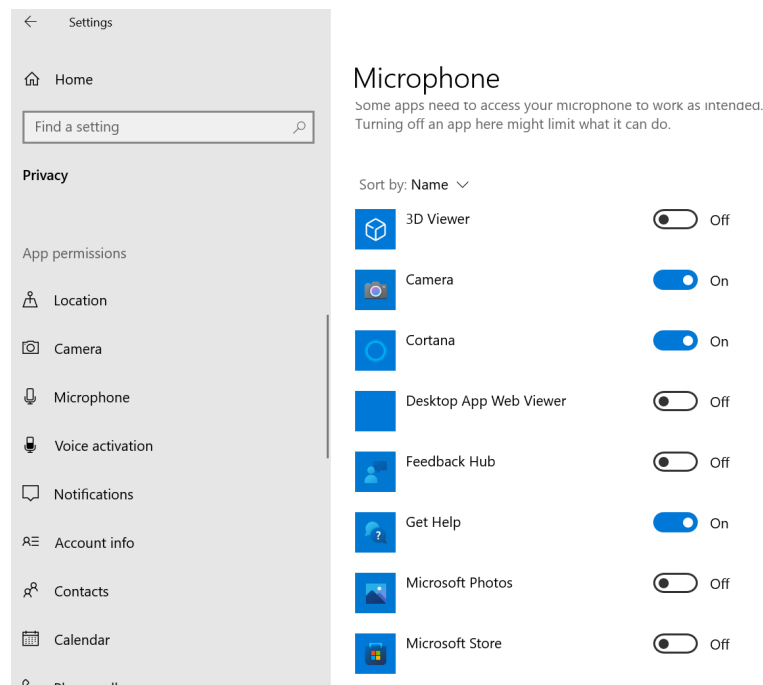
RuntimeBroker.exe

“RuntimeBroker.exe” is an executable which that is located at “%windir%\System32\RuntimeBroker.exe” (On 64 bit systems there is only a 64-bit version with no 32-bit version—in contrast to other executables such as cmd.exe).

“RuntimeBroker.exe” is running the permissions of the user (from whom the session was created). “RuntimeBroker.exe” is triggered from execution if the Windows Store is opened or any installed UWP app is started. By the way UWP apps are also known as Windows App/Windows Store App/Metro App³⁹.

Overall, “RuntimeBroker.exe” is responsible for managing the permissions for “Windows Store App”. We can think about it as a middleman between the application and operating system capabilities⁴⁰.

Thus, when an UWP application tries to access a specific OS resource “RuntimeBroker.exe” checks if the application has the appropriate permissions for that. In case it does not, “RuntimeBroker.exe” can ask the user to grant the permissions. We can modify the permissions for different applications using the “Settings” screen (Privacy->App permissions) - as shown in the screenshot below.



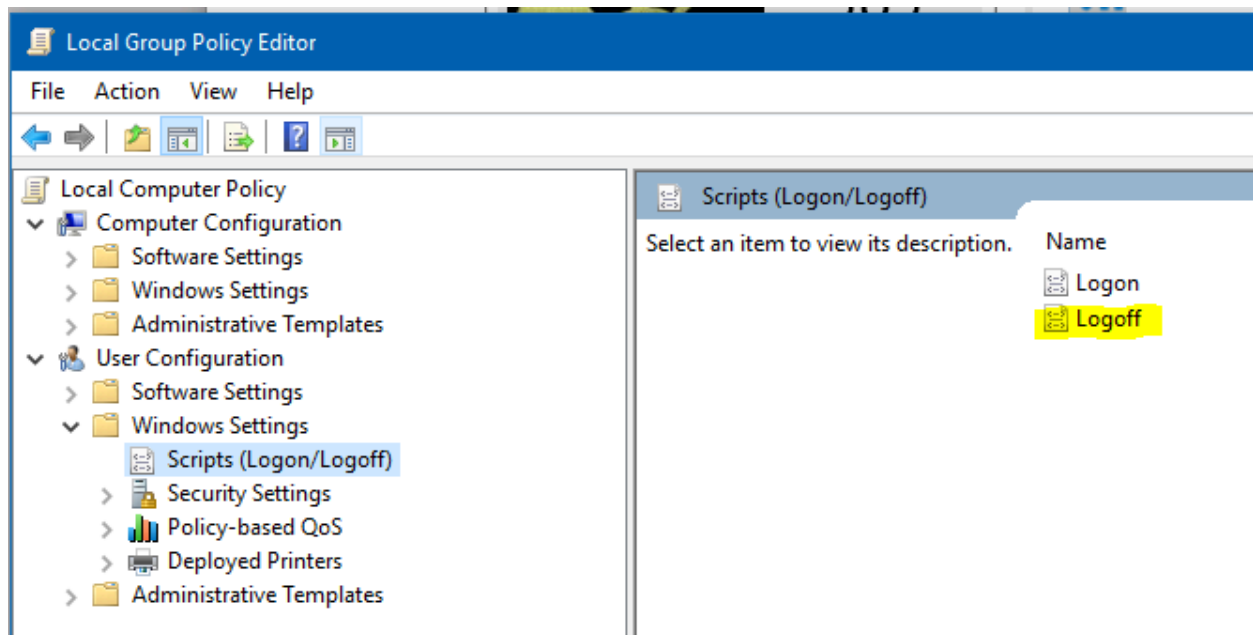
³⁹ <https://www.file.net/process/runtimebroker.exe.html>

⁴⁰ <https://support.microsoft.com/en-us/windows/runtime-broker-is-using-too-much-memory-ca6ed4e3-2a36-964c-4d2e-8c93980d8a98>

logoff.exe (Session Logoff Utility)

“logoff.exe” (Session Logoff Utility) is a command line tool that allows logging off a user from a session. The session could be the current session in which the command is executed, a specific session identified by a number or a remote session on a different server⁴¹. The executable file is located at “%windir%\System32\logoff.exe”.

Moreover, an administrator can set a script/executable to be executed when the user is logging off. This setting can be configured using a local policy/group policy and is called “Logoff script”. Alos, this configuration is part of the “User Configuration -> Windows Settings -> Scripts” - as shown in the screenshot below⁴². Lastly, we can also go over a reference code for “logoff.exe” from ReactOS⁴³.



⁴¹ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/logoff>

⁴² <https://social.technet.microsoft.com/Forums/en-US/f9f011e2-59fc-42d3-a1a4-251536ce8287/i-need-to-automatically-run-an-app-at-logoff?forum=win10itprosetup>

⁴³ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/applications/logoff>

cscript.exe (Microsoft ® Console Based Script Host)

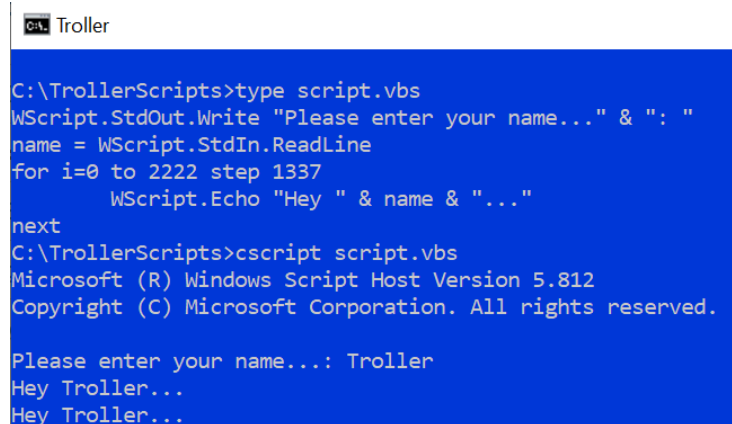
“cscript.exe” is the “Microsoft ® Console Based Script Host” which is a command line version of the “Windows Script Host”. It also allows setting script properties using command line options⁴⁴.

Also, “cscript.exe” is a PE binary file located at “%windir%\System32\cscript.exe”. On a 64-bit system (with a 64-bit OS installed) there is also a 32-bit based version located at “%windir%\SysWOW64\cscript.exe”.

Overall, the “Windows Script Host” (WSH) is an automation technology that enables scripting which was first introduced in Windows 95 (after build 950a) and became a standard component since Windows 98 (build 1111). It has support for different language engines, by default it supports JScript (*.js/*.jse) and VBScript (*.vbs/*.vbe) out of the box⁴⁵.

Moreover, users can also install other scripting engines for WSH like Perl and Python . By using WSH we can also leverage COM (). In VBScript we can do so by calling CreateObject() and in JScript we can use an ActiveXObject or call WScript.CreateObject()⁴⁶.

When using “cscript.exe” to run a script to run in a command-line environment we don’t have to use administrator permissions. Alos, “cscript.exe” has multiple command line options for different usages like: interactive mode, debugging mode, passing arguments to the script and more⁴⁷. Lastly, in order to demonstrate the usage of “cscript.exe” I have created a simple script and executed it - as shown in the screenshot below. We can also go over a reference implementation of “cscript.exe” for RactOS⁴⁸.



```
Ca. Troller
C:\TrollerScripts>type script.vbs
WScript.Stdout.Write "Please enter your name..." & ": "
name = WScript.StdIn.ReadLine
for i=0 to 2222 step 1337
    WScript.Echo "Hey " & name & "..."
next
C:\TrollerScripts>cscript script.vbs
Microsoft (R) Windows Script Host Version 5.812
Copyright (C) Microsoft Corporation. All rights reserved.

Please enter your name...: Troller
Hey Troller...
Hey Troller...
```

⁴⁴[https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490887\(v=technet.10\)?redirectedfrom=MSDN](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-xp/bb490887(v=technet.10)?redirectedfrom=MSDN)

⁴⁵ https://en.wikipedia.org/wiki/Windows_Script_Host

⁴⁶ <https://learn.microsoft.com/vi-vn/windows/win32/com/using-com-objects-in-windows-script-host>

⁴⁷ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/cscript>

⁴⁸<https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/applications/cmdutils/cscript>

wscript.exe (Microsoft ® Windows Based Script Host)

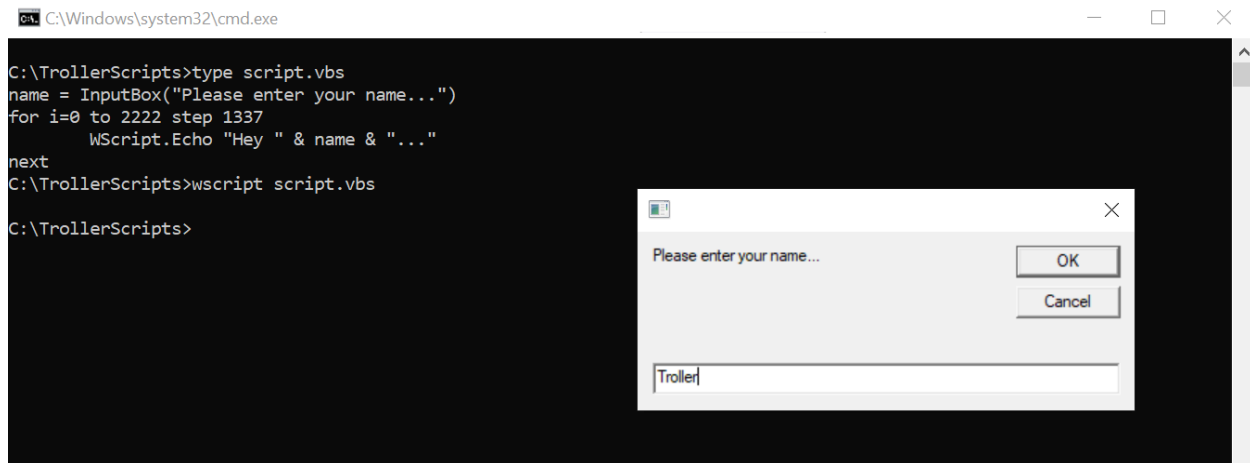
“wscript.exe” is the “Microsoft ® Windows Based Script Host” which provides an environment for executing scripts in a variety of languages⁴⁹. It also allows setting script properties using command line options⁵⁰.

Overall, the “Windows Script Host” (WSH) is an automation technology that enables scripting which was first introduced in Windows 95 (after build 950a) and became a standard component since Windows 98 (build 1111). It has support for different language engines, by default it supports JScript (*.js/*.jse) and VBScript (*.vbs/*.vbe) out of the box⁵¹.

Also, “wscript.exe” is a PE binary file located at “%windir%\System32\wscript.exe”. On a 64-bit system (with a 64-bit OS installed) there is also a 32-bit based version located at “%windir%\SysWOW64\wscript.exe”.

“wscript.exe” allows running the scripts in GUI mode in contrast to “cscript” which is CLI mode⁵². Gui mode means that graphical components could be displayed as the script is being executed - as shown in the screenshot below.

Lastly, in case you want to see a reference implementation of “wscript.exe” I suggest going over the implementation which is part of ReactOS⁵³.



⁴⁹[https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh875526\(v=ws.11\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/hh875526(v=ws.11))

⁵⁰<https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/wscript>

⁵¹https://en.wikipedia.org/wiki/Windows_Script_Host

⁵²<https://medium.com/@boutnaru/the-windows-process-journey-cscript-exe-microsoft-console-based-script-host-5878ba9354a0>

⁵³<https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/applications/cmdutils/wscript>

utilman.exe (Utility Manager)

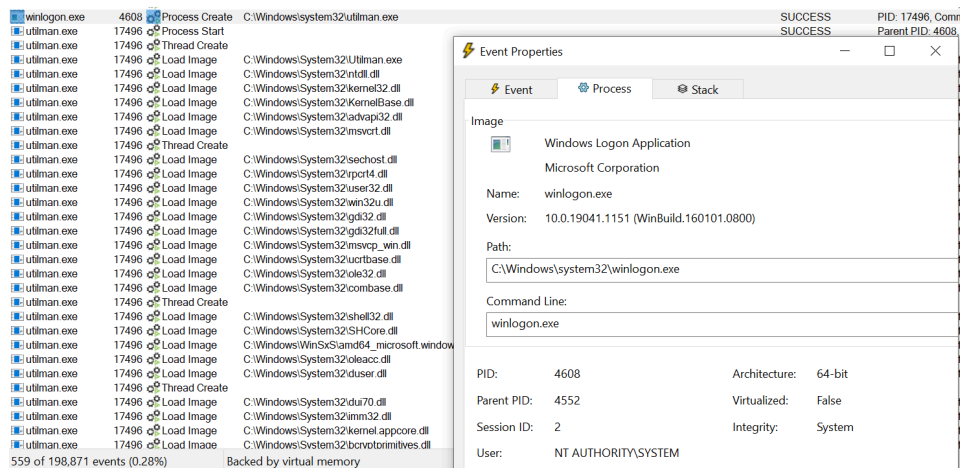
“utilman.exe” is the “Utility Manager” which is a PE binary file located at “%windir%\System32\utilman.exe”. On 64-bit systems there is also a 32-bit version located on “%windir%\SysWOW64\utilman.exe”.

Overall, “utilman.exe” can be started by clicking the icon of “Ease of Access” or by using the keyboard shortcut “WinKey+U”. When using one of those methods while the computer is locked, “utilman.exe” is started by “winlogon.exe” with the permissions of the “LocalSystem” - as shown in the screenshot below. By the way, due to the high level of permissions in use replacing “utilman.exe” is a common trick in order to reset the administrator password in Windows⁵⁴.

Moreover, “utilman.exe” allows accessing the following capabilities: narrator, magnifier, onscreen keyboard, high contrast, sticky keys and filter keys. Narrator is the screen reading application made for blind/visually impaired users⁵⁵. Magnifier is an application that allows users to enlarge the screen content⁵⁶.

Also, sticky keys allows users to use modifier keys (like Ctrl, Shift, Alt and WinKey) without the need of pressing them constantly⁵⁷. Filter keys is a feature that adjusts the keyboard response and ignores repeated keystrokes caused by inaccurate or slow finger movements⁵⁸.

Lastly, in case you want to see a reference implementation of “osk.exe” I suggest going over the implementation which is part of ReactOS⁵⁹.



⁵⁴ <https://learn.microsoft.com/en-us/answers/questions/187973/windows-recovery-cmd>

⁵⁵ <https://support.microsoft.com/en-us/windows/complete-guide-to-narrator-e4397a0d-ef4f-b386-d8ae-c172f109bdb1>

⁵⁶ <https://support.microsoft.com/en-us/windows/use-magnifier-to-make-things-on-the-screen-easier-to-see-414948ba-8b1c-d3bd-8615-0e5e32204198>

⁵⁷ <https://geekflare.com/using-sticky-keys-in-windows/>

⁵⁸ <https://helpdeskgeek.com/how-to/what-are-filter-keys-and-how-to-turn-them-off-in-windows/>

⁵⁹ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/applications/utilman>

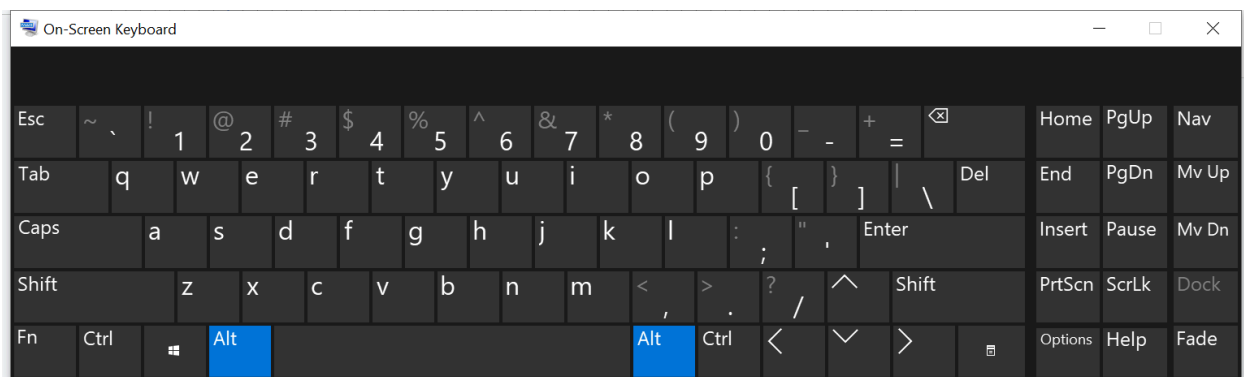
osk.exe (Accessibility On-Screen Keyboard)

“osk.exe” is the “Accessibility On-Screen Keyboard” which presents a virtual keyboard layout inside a resizable window - as shown in the screenshot below. The virtual keyboards enable the user clicking/hovering/scanning using a mouse/joystick in order to select/activate keys⁶⁰.

Moreover, “osk.exe” has a 101/102/106 key layout. “osk.exe” is a PE binary located at “%windir%\System32\osk.exe”. It is bundled with Windows and can provide some features for users with limited mobility⁶¹.

Thus, we don’t need a touch screen in order to interact with “osk.exe”⁶². By the way, “osk.exe” is not the only virtual keyboard available as part of Windows, there is also “TabTip.exe” - but more on there is a separate writeup.

Lastly, in case you want to see a reference implementation of “osk.exe” I suggest going over the implementation which is part of ReactOS⁶³.



⁶⁰ <https://www.file.net/process/osk.exe.html>

⁶¹ <https://www.processlibrary.com/en/directory/files/osk/21965/>

⁶² <https://support.microsoft.com/en-us/windows/use-the-on-screen-keyboard-osk-to-type-ecbb5e08-5b4e-d8c8-f794-81dbf896267a>

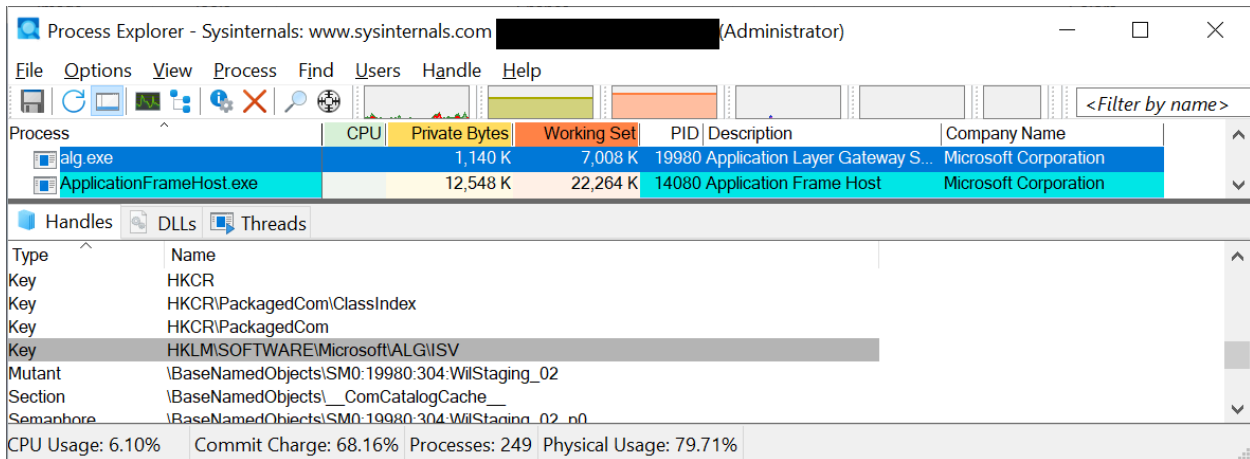
⁶³ <https://github.com/reactos/reactos/tree/47f3a4e144b897da0e0e8cb08c2909645061dec9/base/applications/osk>

alg.exe (Application Layer Gateway Service)

“alg.exe” is the “Application Layer Gateway Service” (ALG) which is configured as a Windows service. Based on the description of the service it provides support for 3rd party protocol plug-ins for Internet Connection Sharing (ICS). The service is executed with the permission of the “LocalService” user. “alg.exe” is a PE binary which is stored in the following location: “%windir%\System32\alg.exe”.

Generally, an “Application Layer Gateway” (ALG) allows a gateway to parse payloads and take actions such as allow/drop/other based on the data contained in the payloads⁶⁴. Thus, ALG’s plugins can modify data in packets, think about things like IP addresses and port numbers⁶⁵.

Lastly, “alg.exe” is started by “services.exe” with the permission of “NT AUTHORITY\LOCAL SERVICE” user. There should be at most only one instance of “alg.exe”. “alg.exe” parses information about supported plugins from “HKLM\SOFTWARE\Microsoft\ALG\ISV”⁶⁶. We can see in the screenshot below that there is a handle to that registry location.



⁶⁴ <https://www.juniper.net/documentation/us/en/software/junos/alg/alg.pdf>

⁶⁵ https://en.wikipedia.org/wiki/Application-level_gateway

⁶⁶ https://www.sigma-uk.net/tech/windows_ftp_alg_iis

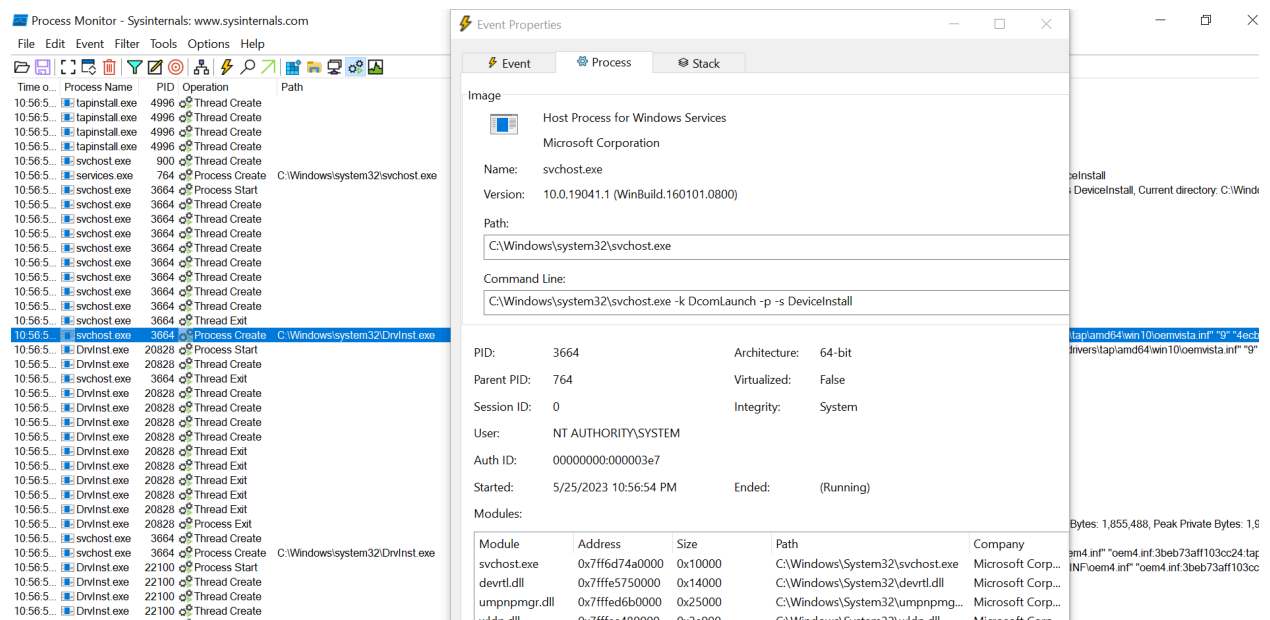
DrvInst.exe (Driver Installation Module)

“DrvInst.exe” is a PE executable located at “%windir%\System32\drvinst.exe”, it is known as “Driver Installation Module”. Since Windows Vista when PnP (Plug and Play) manager detects a new device “DrvInst.exe” is started. It is used for searching and installing the relevant driver for the new device detected⁶⁷.

“DrvInst.exe” can also be used for installing drivers while installing a software package. Let us take for example the installation of “OpenVPN Connect”⁶⁸.

Thus, as with most VPN (Virtual Private Network) solutions there is a need to install a TAP driver, which is a virtual network device⁶⁹. This causes “services.exe” to launch a new process using the following arguments “C:\Windows\system32\svchost.exe -k DcomLaunch -p -s DeviceInstall”, which is part of the “DCOM Server Process Launcher”. It is executed with the permission of the “LocalSystem” user.

Moreover, by passing as an argument “DeviceInstall” “svchost.exe” loads “%windir%\System32\umpnpgm.dll”, which is the “User-mode Plug-and-Play Service”. This instance of “svchost.exe” is the one that starts “DrvInst.exe”. It also loads “%windir%\System32\devrtl.dll” (Device Management Run Time Library) - as shown in the screenshot below.



⁶⁷<https://learn.microsoft.com/en-us/windows-hardware/drivers/install/debugging-device-installations-with-a-user-mode-debugger>

⁶⁸<https://openvpn.net/client/>

⁶⁹<https://www.techradar.com/vpn/what-is-a-tap-adapter>

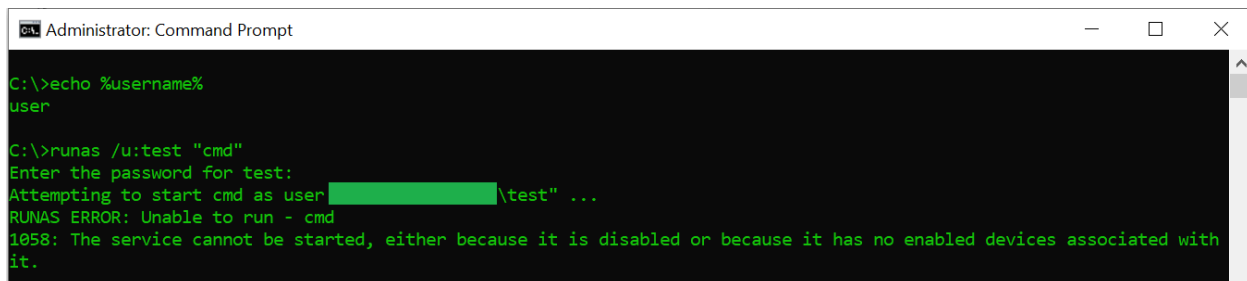
runas.exe (Run As Utility)

“runas.exe” is an executable aka “Run As Utility”, which is located at “%windir%\System32\runas.exe”. On 64 bit systems there is also a 32-bit version located at “%windir%\SysWow64\runas.exe”.

Overall, “runas.exe” allows a user to execute specific programs/tools with different permissions than the logged-on user. “runas.exe” also has multiple parameters that can be used like passing credentials from a smartcard instead of a password, loading the user’s profile and more⁷⁰.

Moreover, “runas.exe” is dependent on the “Secondary Logon” service. The description of the service states that it “enables starting processes under alternate credentials. If this service is stopped, this type of logon access will be unavailable. If this service is disabled, any services that explicitly depend on it will fail to start”. As described if the service is disabled “runas.exe” will fail - as shown in the screenshot below.

Thus, in case the “Secondary Logon” service can be started it is done with the following command line: “%windir%\system32\svchost.exe -k netsvcs -p -s seclogon” with the permissions of the “Local System” user. Also, in this case “svchost.exe” will load “%windir%\System32\seclogon.dll” (Secondary Logon Service DLL).



```
Administrator: Command Prompt
C:\>echo %username%
user
C:\>runas /u:test "cmd"
Enter the password for test:
Attempting to start cmd as user [REDACTED]\test" ...
RUNAS ERROR: Unable to run - cmd
1058: The service cannot be started, either because it is disabled or because it has no enabled devices associated with it.
```

⁷⁰[https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc771525\(v=ws.11\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc771525(v=ws.11))

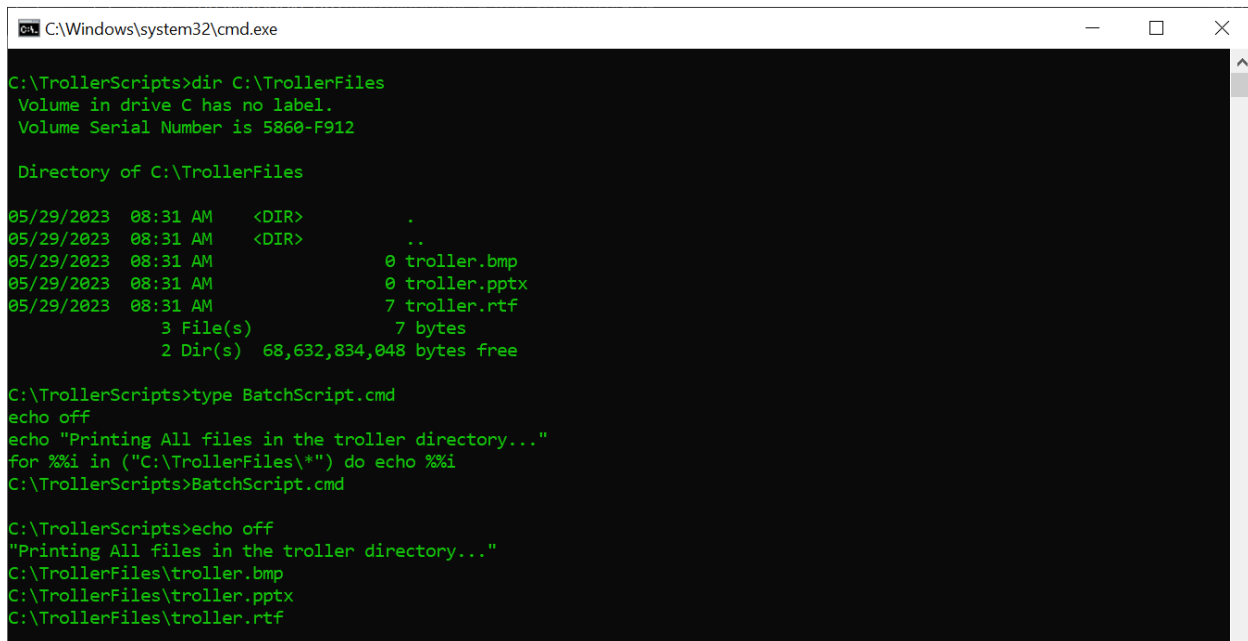
cmd.exe (Windows Command Processor)

“cmd.exe” is the “Windows Command Processor” which is the default CLI (command line interface/interpreter) of Windows (and also reactOS). By the way, it is also known as “Command Prompt”. It is the replacement of “command.com” which was relevant from MS-DOS to Windows XP. In Windows NT/Windows 2000 and Windows XP there was both “cmd.exe” and “command.com”⁷¹.

The executable is located at “%windir%\System32\cmd.exe”. On 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\cmd.exe”. Also, “cmd.exe” allows the execution of any script/executable installed on the system or one of the internal command which included as part of “cmd.exe” like: “cd”, “copy” and “md”⁷².

Moreover, “cmd.exe” supports executing batch scripts - as shown in the screenshot below. I suggest going through “Windows Batch Scripting” for more information⁷³.

Lastly, for a reference of “cmd.exe” I suggest going over the implementation of “cmd.exe” as part of ReacOS⁷⁴.



```
C:\Windows\system32\cmd.exe
C:\TrollerScripts>dir C:\TrollerFiles
Volume in drive C has no label.
Volume Serial Number is 5860-F912

Directory of C:\TrollerFiles

05/29/2023  08:31 AM  <DIR>          .
05/29/2023  08:31 AM  <DIR>          ..
05/29/2023  08:31 AM                0 troller.bmp
05/29/2023  08:31 AM                0 troller.pptx
05/29/2023  08:31 AM                7 troller.rtf
           3 File(s)                7 bytes
           2 Dir(s) 68,632,834,048 bytes free

C:\TrollerScripts>type BatchScript.cmd
echo off
echo "Printing All files in the troller directory..."
for %i in ("C:\TrollerFiles\*") do echo %i
C:\TrollerScripts>BatchScript.cmd

C:\TrollerScripts>echo off
"Printing All files in the troller directory..."
C:\TrollerFiles\troller.bmp
C:\TrollerFiles\troller.pptx
C:\TrollerFiles\troller.rtf
```

⁷¹ <https://www.computerhope.com/cmd.htm>

⁷² <https://wishmesh.com/2014/09/ms-dos-cmd-exe-command-prompt-cd-md-copy/>

⁷³ https://en.wikibooks.org/wiki/Windows_Batch_Scripting

⁷⁴ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aecaf4515603f3f/base/shell/cmd>

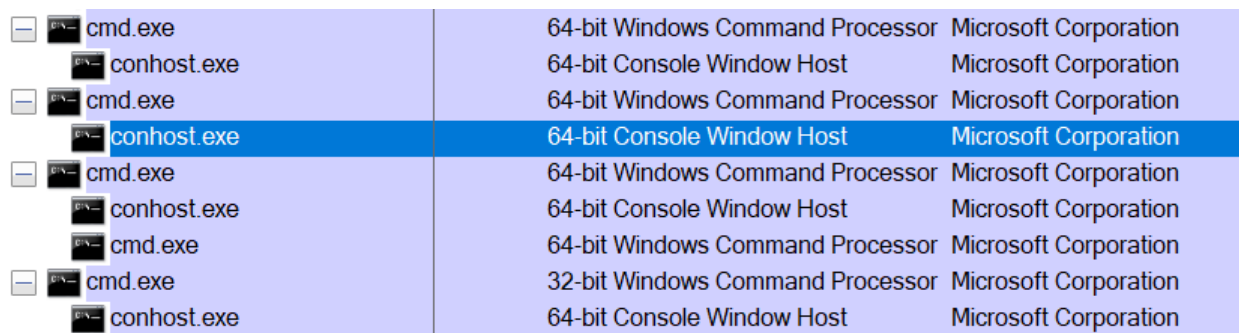
conhost.exe (Console Window Host)

“conhost.exe” is an executable aka the “Console Window Host”, which is located at “%windir%\System32\conhost.exe”. The goal of “conhost.exe” is to provide an interface between “cmd.exe”⁷⁵ and “explorer.exe”⁷⁶.

Thus, “conhost.exe” is both the server application (for Windows Console API) and also the classic Windows user interface for working with CLI (command line interface) application. Historically, those were the job of “csrss.exe”⁷⁷ but they were extracted for isolation and security reasons⁷⁸.

Moreover, one of the duties of “conhost.exe” is to provide the ability to “drag and drop” folders/files into “cmd.exe”. By the way, every 3rd party application can use “conhost.exe”⁷⁹. When “conhost.exe” is started with the permissions of the user which “cmd.exe” was started with.

Lastly, we can have multiple instances of “conhost.exe”. For each instance of “cmd.exe” (which is not a descendant of another “cmd.exe”) there will be an instance of “conhost.exe”. Also, in case of a 64-bit system even if a 32-bit “cmd.exe” an instance of a 64-bit “conhost.exe” is going to be started. A demonstration of those points is shown in the screenshot below (taken using “Process Explorer”).



cmd.exe	64-bit Windows Command Processor	Microsoft Corporation
conhost.exe	64-bit Console Window Host	Microsoft Corporation
cmd.exe	64-bit Windows Command Processor	Microsoft Corporation
conhost.exe	64-bit Console Window Host	Microsoft Corporation
cmd.exe	64-bit Windows Command Processor	Microsoft Corporation
conhost.exe	64-bit Console Window Host	Microsoft Corporation
cmd.exe	64-bit Windows Command Processor	Microsoft Corporation
conhost.exe	64-bit Console Window Host	Microsoft Corporation
cmd.exe	32-bit Windows Command Processor	Microsoft Corporation
conhost.exe	64-bit Console Window Host	Microsoft Corporation

⁷⁵ <https://medium.com/@boutnaru/the-windows-process-journey-cmd.exe-windows-command-processor-501be17ba81b>

⁷⁶ <https://medium.com/@boutnaru/the-windows-process-journey-explorer.exe-windows-explorer-9a96bc79e183>

⁷⁷ <https://medium.com/@boutnaru/the-windows-process-journey-csrss.exe-client-server-runtime-subsystem-cb5fa34c47db>

⁷⁸ <https://learn.microsoft.com/en-us/windows/console/definitions>

⁷⁹ <https://www.lifewire.com/conhost-exe-4158039>

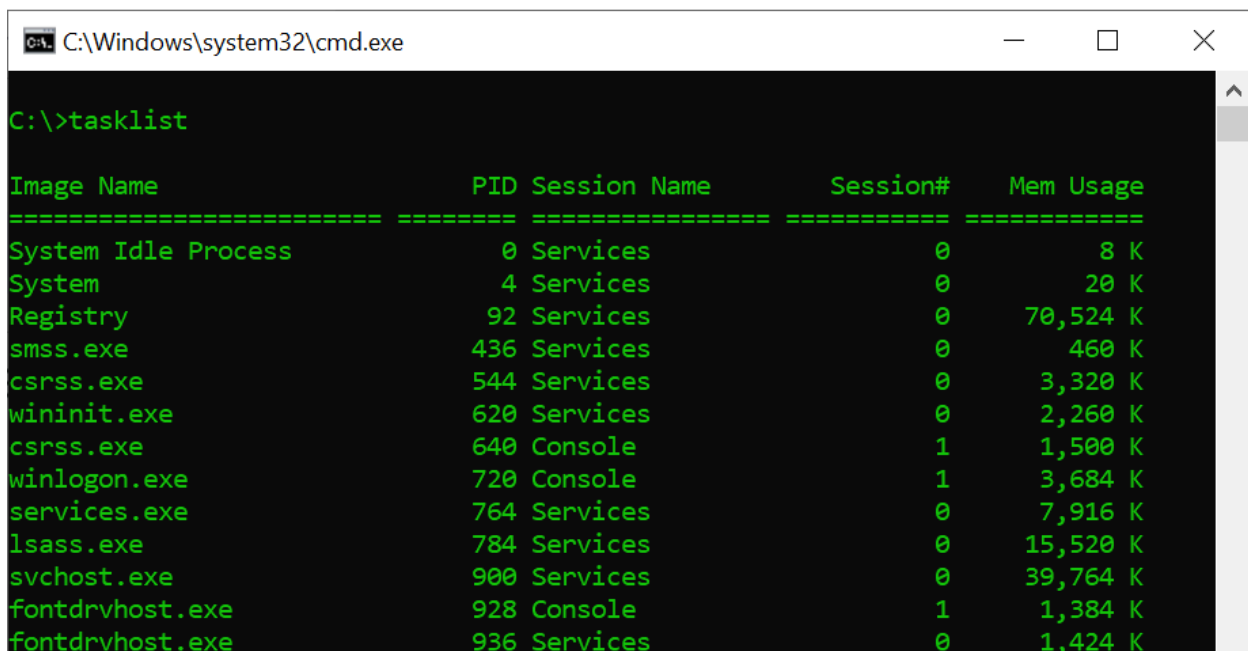
tasklist.exe (Lists the Current Running Tasks)

“tasklist.exe” is an executable which is located at “%windir%\System32\tasklist.exe”. It allows displaying the list of currently running processes on the system⁸⁰. On 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\tasklist.exe”.

Moreover, a user with sufficient permissions can also list the processes of a remote system using “tasklist.exe” by using the “/s” command line switch. For more information about the other switches which are available please refer to <https://ss64.com/nt/tasklist.html>.

Overall, a user can display the following attributes for each displayed process: image name, pid, session number, session name, cpu time, memory usage, user name, service name (if relevant), window title (if relevant) and more.

Lastly, for a reference of “cmd.exe” I suggest going over the implementation of “cmd.exe” as part of ReacOS⁸¹.



```
C:\Windows\system32\cmd.exe
C:\>tasklist

Image Name                PID Session Name        Session#    Mem Usage
=====
System Idle Process        0 Services            0           8 K
System                     4 Services            0          20 K
Registry                   92 Services            0       70,524 K
smss.exe                   436 Services            0          460 K
csrss.exe                  544 Services            0       3,320 K
wininit.exe                620 Services            0       2,260 K
csrss.exe                  640 Console              1       1,500 K
winlogon.exe               720 Console              1       3,684 K
services.exe               764 Services            0       7,916 K
lsass.exe                  784 Services            0      15,520 K
svchost.exe                900 Services            0      39,764 K
fontdrvhost.exe           928 Console              1       1,384 K
fontdrvhost.exe           936 Services            0       1,424 K
```

⁸⁰ [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc730909\(v=ws.11\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/cc730909(v=ws.11))

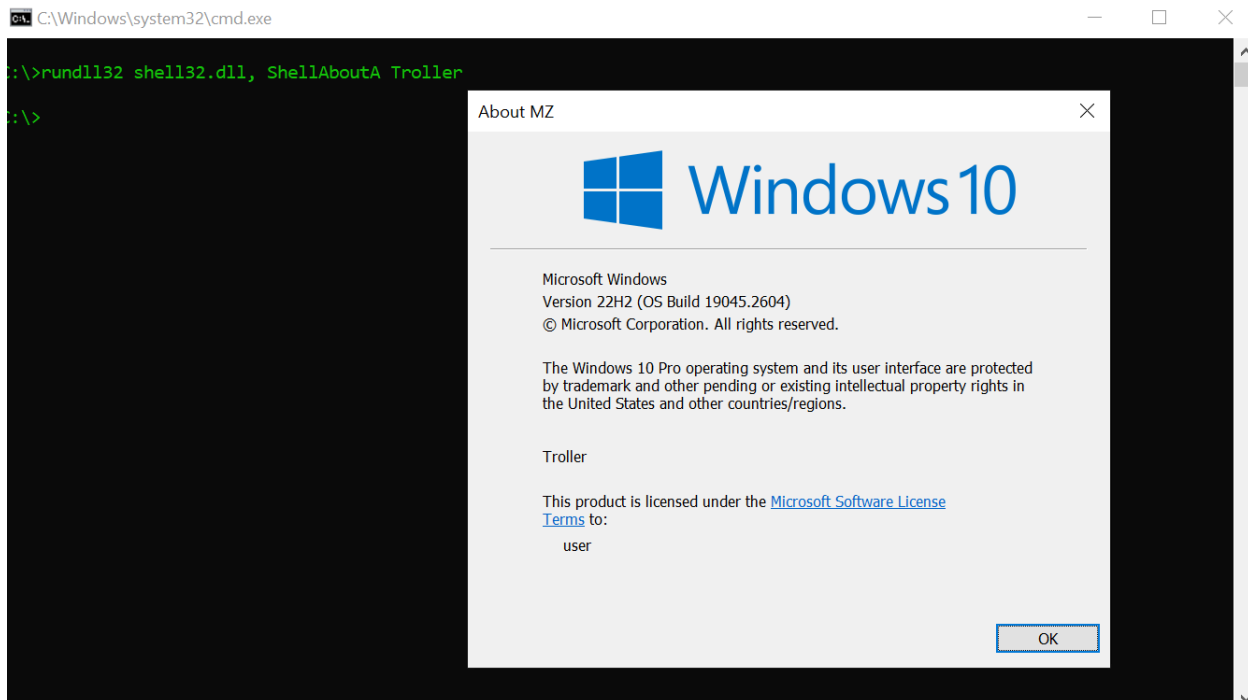
⁸¹ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/applications/cmdutils/tasklist>

rundll32.exe (Windows Host Process)

“rundll32.exe” is an executable aka the “Windows Host Process” (based on the description field of the PE file), which is located at “%windir%\System32\rundll32.exe”. On a 64 bit-system the file still has the same name (including the number 32) and a 32-bit version is located at “%windir%\SysWOW64\rundll32.exe”.

Overall, the goal of “rundll32.exe” is to load a DLLs (Dynamic Link Libraries) and run a functionality stored in those files⁸². The DLLs are loaded using “LoadLibraryExW”⁸³. “rundll32.exe” is digitally signed by Microsoft and shipped by default with the operating system. By the way, there are also places that say “rundll32.exe” means “Run a DLL as an App”⁸⁴.

The way is which we can call a function from a “*.dll” file is by passing the name of the file and the name of the function. We can also pass arguments to a function while using “rundll32.exe”⁸⁵. An example of using “rundll32.exe” is shown in the screenshot below. Also, for more examples of using “rundll32.exe” I suggest going over the following link <https://www.thewindowsclub.com/rundll32-shortcut-commands-windows>. Lastly, for an implementation reference of “rundll32.exe” I suggest going over the one in ReacOS⁸⁶.



⁸² <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/rundll32>

⁸³ <https://www.cybereason.com/blog/rundll32-the-infamous-proxy-for-executing-malicious-code>

⁸⁴ <https://www.file.net/process/rundll32.exe.html>

⁸⁵ <https://stmxcsl.com/micro/rundll-parse-args.html>

⁸⁶ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/system/rundll32>

net.exe (Network Command)

“net.exe” is the “Net Command” which is a command line that allows managing different aspects of the operating system such as: users, groups, services and network connections⁸⁷. Also, “net.exe” is a PE binary file located at “%windir%\System32\net.exe” which is signed by Microsoft. On 64-bit based versions of Windows there is also a 32-bit version of the binary located at “%windir%\SysWOW64\net.exe”.

Overall, they are 19 sub commands in net: “accounts”, “computer”, “config”, “continue”, “file”, “group”, “help”, “helpmsg”, localgroup”, “pause”, “session”, “share”, “start”, “statistics”, “stop”, “time”, “use”, “user” and “view”. By using “net help” we can get an explanation about each sub command. In the table below I have gathered a short description for each sub command (excluding “net help”). Lastly, we can also go over a reference implementation of “net.exe” from ReacOS⁸⁸.

net command	description
net accounts	updates the user accounts database and modifies password and logon requirements for all accounts
net computer	adds or deletes computers from a domain database
net config	displays configuration information of the Workstation or Server service
net continue	reactivates a Windows service that has been suspended by “net pause”
net file	closes a shared file and removes file locks. When used without options, it lists the open files on a server.
net group	adds, displays, or modifies global groups on servers (used on an AD environment)
net helpmsg	displays information about Windows network messages (such as error, warning, and alert messages)
net localgroup	modifies local groups on computers. When used without options, it displays the local groups on the computer
net pause	suspends a Windows service or resource. Pausing a service puts it on hold
net session	lists or disconnects sessions between the computer and other computers on the network. When used without options, it displays information about all sessions with the computer of current focus
net share	makes a server's resources available to network users. When used without options, it lists information about all resources being shared on the computer
net start	lists running services, also can start a specific service
net statistics	Displays the statistics log for the local Workstation service
net stop	Stopping a service cancels any network connections the service is using
net time	synchronizes the computer's clock with that of another computer or domain, or displays the time for a computer or domain. When used without options displays the current date and time at the computer
net use	connects a computer to a shared resource or disconnects a computer from a shared resource. When used without options, it lists the computer's connections
net user	creates and modifies user accounts on computers. When used without switches, it lists the user accounts for the computer
net view	displays a list of resources being shared on a computer. When used without options, it displays a list of computers in the current domain or Network

⁸⁷ <https://attack.mitre.org/software/S0039/>

⁸⁸ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/applications/network/net>

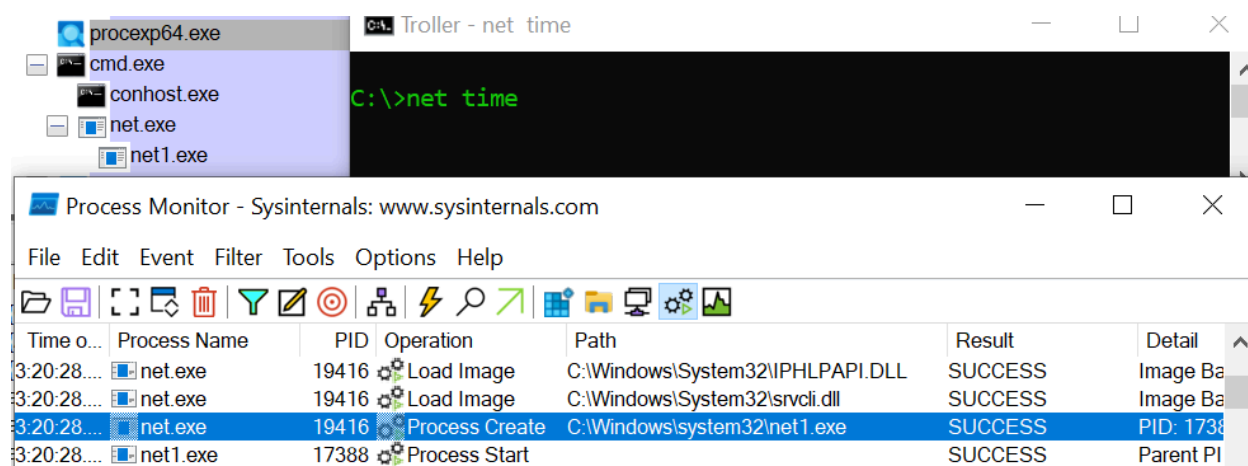
net1.exe (Net Command for the 21st Century)

“net1.exe” is known as the “Net Command for the 21st Century”⁸⁹. It is a PE binary file that is signed by Microsoft, which is located at “%windir%\system32\net1.exe”. On 64-bit versions of Windows there is also a 32-bit version of the file located at “%windir%\SysWOW64\net1.exe”.

Overall, the “net1.exe” was created as a temporary fix for the Y2K problem that affected “net.exe”⁹⁰. There was an issue while using the command “net user [USERNAME] /times” which is responsible for configuring the logon hours of the user⁹¹.

Thus, “net1.exe” is executed for specific functionality when “net.exe” is run⁹². For example when calling “net time” an instance of “net1.exe” is started by “net.exe” using the command “net1 time” - as seen in the screenshot below.

Lastly, “net1.exe” supports every command the “net.exe” supports. The issue with “net.exe” was corrected in Windows XP, however “net1.exe” is still available today for backward compatibility with old scripts that might use it⁹³.



⁸⁹ https://www.file.net/process/net1_exe.html

⁹⁰ <https://www.lifewire.com/net-command-2618094>

⁹¹ <https://web.archive.org/web/20140830150320/http://support.microsoft.com/kb/240195>

⁹² <https://attack.mitre.org/software/S0039/>

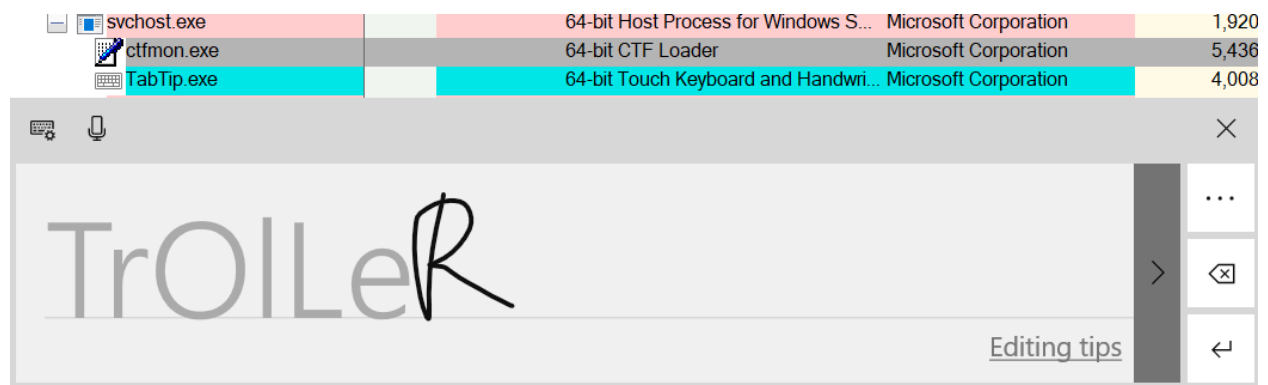
⁹³ <https://ss64.com/nt/net.html>

TabTip.exe (Touch Keyboard and Handwriting Panel)

“TabTip.exe” (Touch Keyboard and Handwriting Panel) is also known as “Tablet Text Input Panel”. It is an interface developed by Microsoft which allows inputting text in different ways: handwriting to text, speech to text and by clicking on the screen like a keyboard⁹⁴.

The usage of “TabTip.exe” as a keyboard is very similar to “osk.exe”⁹⁵. The main goal of “TabTip.exe” is to provide handwriting input. This means that even applications that don’t have this capability can use “TabTip.exe” to provide users with the ability of writing instead of typing⁹⁶ - as shown in the screenshot below.

Overall, “TabTip.exe” is a 64-bit PE binary located at “%ProgramFiles%\Common Files\microsoft shared\ink\TabTip.exe”, which is digitally signed by Microsoft. When “TabTip.exe” is launched it is started as a child process of the service TabletInputService (Touch Keyboard and Handwriting Panel Service), similar to “ctfmon.exe”⁹⁷ - as shown in the screenshot below. This service is hosted by “svchost.exe”⁹⁸, which loads the “%windir%\System32\TabSvc.dll”.



⁹⁴ <https://www.file.net/process/tabtip.exe.html>

⁹⁵ <https://medium.com/@boutnaru/the-windows-process-journey-osk-exe-accessibility-on-screen-keyboard-72823695321e>

⁹⁶ <https://windowsreport.com/tabtip-exe/>

⁹⁷ <https://medium.com/@boutnaru/the-windows-process-journey-ctfmon-exe-ctf-loader-148f10f5401>

⁹⁸ <https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f>

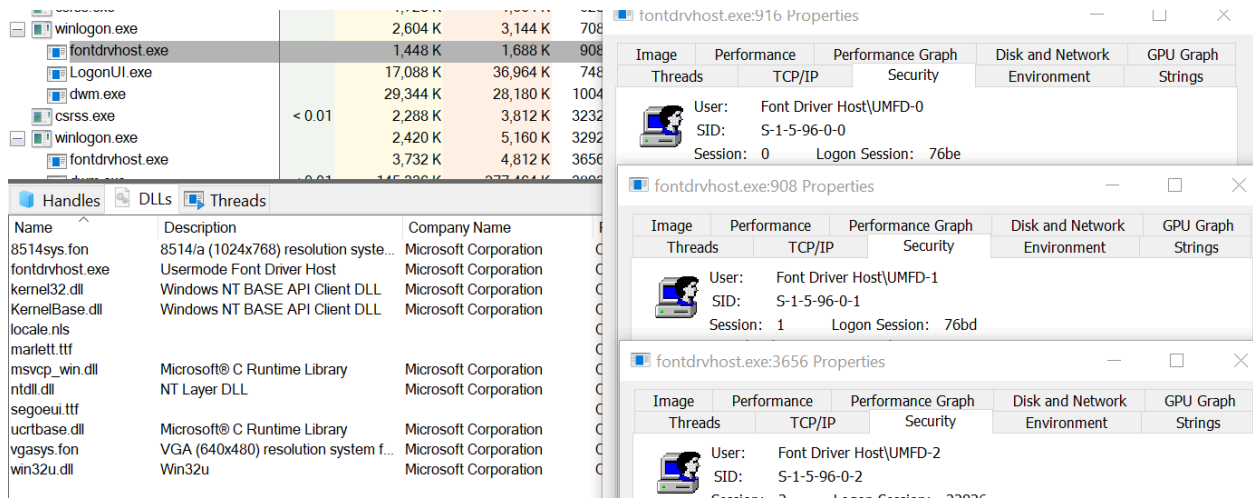
fontdrvhost.exe (Usermode Font Driver Host)

On Windows 8.1 (and previous versions) the parsing of fonts takes place in a kernel driver (atmfd.dll, yes they are DLLs which are executed in kernel mode). This was accessible via graphical syscalls exported by win32k.sys, thus it created an attack surface that could lead to privilege escalation. Thus, from Windows 10 the parsing code was moved to the restricted user-mode process “fontdrvhost.exe”⁹⁹

Overall, “fontdrvhost.exe” is an executable which is located at “%windir%\System32\fontdrvhost.exe“ (On 64-bit systems there is also a 32-bit located at “%windir%\SysWOW64\fontdrvhost.exe”). It is executed with the permissions of a user in the following pattern: “Font Driver Host\UMFD[SessionID]”. Also, the SID of the user is in the pattern of “S-1-5-96-[SessionID]” - as you can see in the screenshot below. Also, “fontdrvhost.exe” is a PE binary that is digitally signed by Microsoft.

Moreover, on session 0 “fontdrvhost.exe” is started by “wininit.exe”¹⁰⁰, in the following sessions (1, 2 , etc) it is started by “winlogon.exe”¹⁰¹. Thus, the number of instances of “fontdrvhost.exe” should be as the number of opened sessions on the Windows system.

Lastly, UMDF stands for “User Mode Driver Framework”, which allows running driver in host processes¹⁰².



⁹⁹ <https://googleprojectzero.blogspot.com/2021/01/in-wild-series-windows-exploits.html>

¹⁰⁰ <https://medium.com/@boutnaru/the-windows-process-journey-wininit-exe-windows-start-up-application-5581bfe6a01e>

¹⁰¹ <https://medium.com/@boutnaru/the-windows-process-journey-winlogon-exe-windows-logon-application-88a1d4d3e13c>

¹⁰² <https://learn.microsoft.com/en-us/windows-hardware/drivers/wdf/user-mode-driver-framework-frequently-asked-questions>

OpenWith.exe (Pick an App)

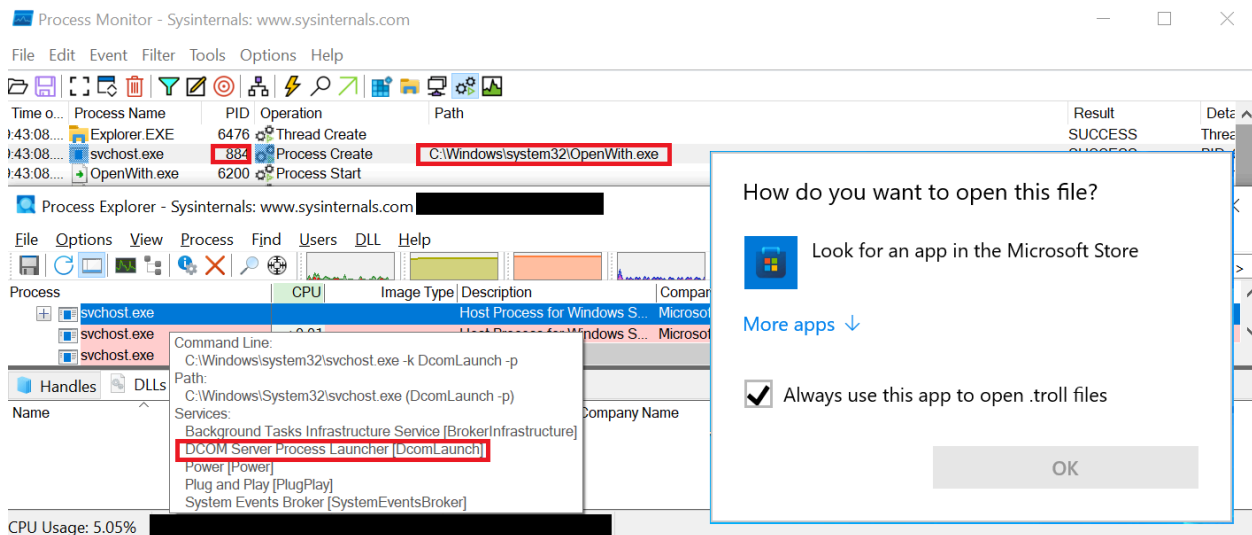
“OpenWith.exe” is also known as the “Pick an App”, it is located at “%windir%\System32\OpenWith.exe” and it is digitally signed by Microsoft. On 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\OpenWith.exe”.

Overall, “OpenWith.exe” is used for selecting the application we want to open a file with a specific extension - as shown in the screenshot below. You might expect that “explorer.exe” is going to start “OpenWith.exe”, however it is done by the “DCOM Server Process Launcher” service which is hosted by “svchost.exe”¹⁰³ - as shown in the screenshot below.

Moreover, due to the reason the hosting “svchost.exe” is running with the permissions of the “LocalSystem” the creation of the “OpenWith.exe” process is done using the API “CreateProcessWithUserW”¹⁰⁴. It allows “svchost.exe” to execute “OpenWith.exe” with the permissions of the logged on user (the same access token as “explorer.exe”).

At the end, when we select an app the next time a double click is identified “explorer.exe”¹⁰⁵ is going to start an instance of the application associated with the extension and pass as an argument the full path of the app.

Thus, if we associate “%windir%\system32\notepad.exe” with “*.troll” a double click on “troller.troll” leads to the following command line to be executed: ““C:\Windows\system32\notepad.exe" C:\Users\[USERNAME]\Desktop\troller.trl”.



¹⁰³<https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f>

¹⁰⁴<https://learn.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessasuserw>

¹⁰⁵<https://medium.com/@boutnaru/the-windows-process-journey-explorer-exe-windows-explorer-9a96bc79e183>

mavinject.exe (Microsoft Application Virtualization Injector)

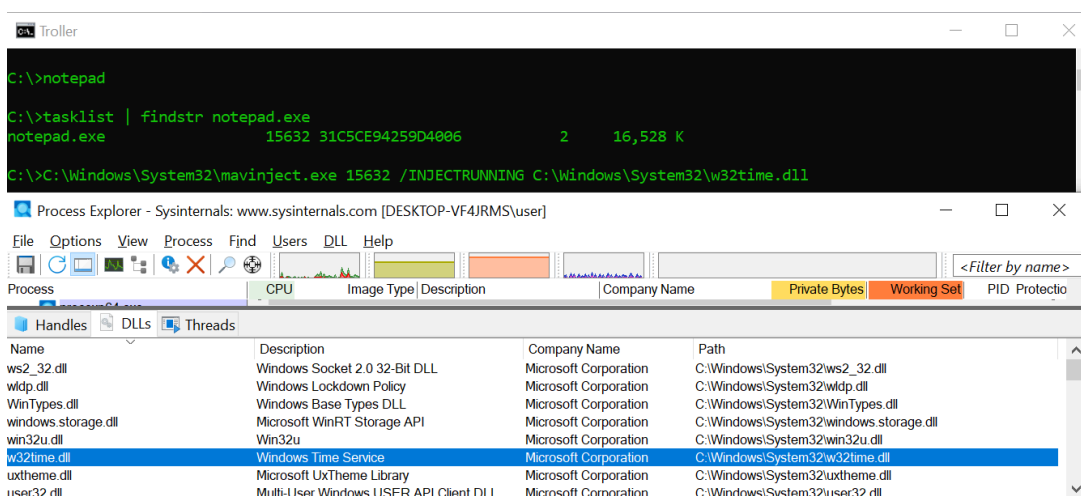
“mavinject.exe” is the “Microsoft Application Virtualization Injector” which is part of App-V (Microsoft Application Virtualization). App-V allows the delivering of applications to users as “virtual applications”. This means that “virtual applications” are installed on a central managed server. They are “streamed” to users as a service as they are needed. From the user’s perspective it acts as an installed application locally¹⁰⁶.

Overall, “mavinject.exe” is a PE binary located at “%windir%\System32\mavinject.exe”, which is digitally signed by Microsoft. In case of a 64-bit system there is also a 32-bit version located at “%windir%\SysWOW64\mavinject.exe”.

Moreover, using “mavinject.exe” we can perform DLL injection, meaning loading a DLL in the address space of a different process. In order to do so we need to run “mavinject.exe” with different arguments like: “mavinject.exe [PID] /INJECTRUNNING [PATH_TO_DLL_TO_LOAD]” - as shown in the screenshot below.

Also, there are other arguments that can be used “/HMODULE” which allows import descriptor injection. We can use it in the following manner: “mavinject.exe PID /HMODULE=BASE_ADDRESS PATH_DLL ORDINAL_NUMBER”¹⁰⁷.

Lastly, “mavinject.exe” uses the following Win32 API calls: VirtualProtectEx, CreateRemoteThread, VirtualAllocEx, OpenProcess, LoadLibraryW and WriteProcessMemory¹⁰⁸.



¹⁰⁶ <https://learn.microsoft.com/en-us/windows/application-management/app-v/appv-about-appv>

¹⁰⁷ <https://unprotect.it/technique/system-binary-proxy-execution-mavinject/>

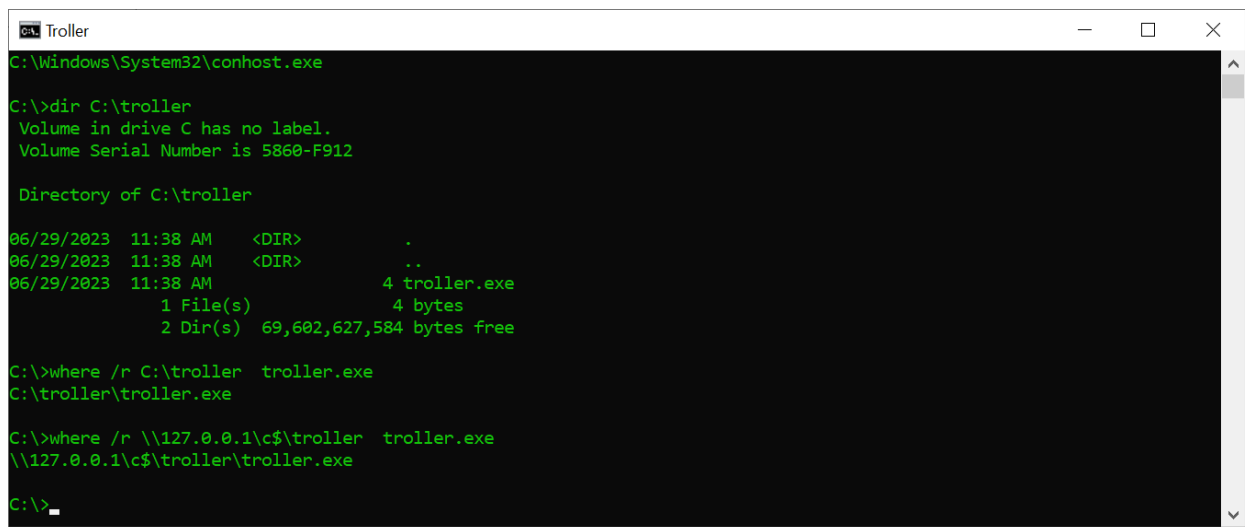
¹⁰⁸ <https://posts.specterops.io/mavinject-exe-functionality-deconstructed-c29ab2cf5c0e>

where.exe (Lists location of Files)

“where.exe” (List Location of Files) is responsible for displaying the location of files which match a specific search pattern. The search is done in the current directory and in the path which are declared as part of the “PATH” environment variable¹⁰⁹. It is equivalent to the “which” command under Linux¹¹⁰.

Overall, “where.exe” is a PE binary file located at “%windir%\System32\where.exe”. On 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\where.exe”. Also, the file is digitally signed by Microsoft.

Moreover, we can use “where.exe” to search in subdirectories from a specific location using the “/r” switch. We can also perform the search remotely by specifying a UNC path¹¹¹ - as shown in the screenshot below.



```
C:\Windows\System32\conhost.exe
C:\>dir C:\troller
Volume in drive C has no label.
Volume Serial Number is 5860-F912

Directory of C:\troller

06/29/2023  11:38 AM  <DIR>          .
06/29/2023  11:38 AM  <DIR>          ..
06/29/2023  11:38 AM                4 troller.exe
               1 File(s)                4 bytes
               2 Dir(s)  69,602,627,584 bytes free

C:\>where /r C:\troller troller.exe
C:\troller\troller.exe

C:\>where /r \\127.0.0.1\c$\troller troller.exe
\\127.0.0.1\c$\troller\troller.exe

C:\>
```

¹⁰⁹ <https://ss64.com/nt/where.html>

¹¹⁰ <https://linux.die.net/man/1/which>

¹¹¹ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/where>

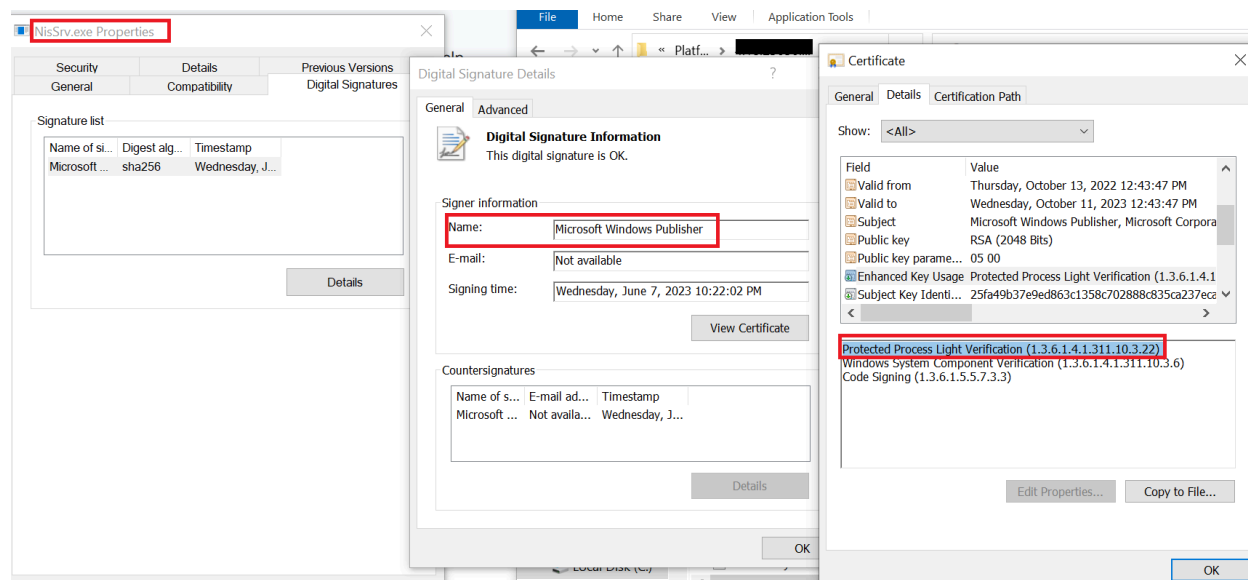
NisSrv.exe (Microsoft Network Realtime Inspection Service)

“NisSrv.exe” is a PE binary which is the main executable that is started by the “WdNisSvc” service aka “Microsoft Network Realtime Inspection”. It is executed by “services.exe” with the permissions of the “NT AUTHORITY\LOCAL SERVICE” user (S-1-5-19). The description of the service states it helps in guarding against intrusion attempts targeting known/newly discovered vulnerabilities in network protocols.

Overall, “NisSrv.exe” monitors and inspects network traffic in real-time. By doing that it searches for suspicious behavior that might suggest an exploit targeting the network protocol is being executed¹¹².

Moreover, “NisSrv.exe” is part of the “Windows Defender” platform, which is Microsoft’s endpoint security platform. “Windows Defender” provides attack surface reduction and next generation protection for both OS level and network based¹¹³.

Lastly, “NisSrv.exe” is a PE binary file located at “%ProgramData%\Microsoft\Windows Defender\Platform\[VERSION]\NisSrv.exe”. It is also signed digitally by Microsoft the same way as the main process of “Windows Defender” (MsMpEng.exe), with a signed level of Antimalware (PsProtectedSignerAntimalware-Light) - as shown in the screenshot below.



¹¹²<https://www.howtogeek.com/357184/what-is-microsoft-network-realtime-inspection-service-nissrv.exe-and-why-is-it-running-on-my-pc/>

¹¹³<https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-endpoint?view=o365-worldwide>

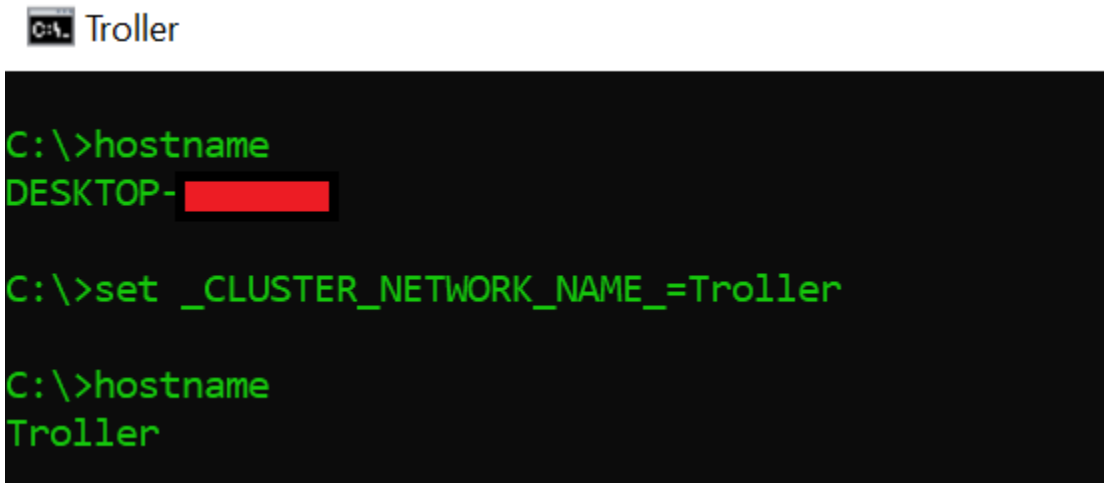
Hostname.exe (Hostname APP)

“hostname.exe” is an executable located at “%windir%\System32\HOSTNAME.EXE”. On a 64-bit system there is also a 32-bit version located at “%windir%\SysWOW64\HOSTNAME.EXE”. The executable is digitally signed by Microsoft.

Overall, “hostname.exe” is responsible for displaying the host name portion of the full computer name. By the way, printing the environment variable %COMPUTERNAME% will output the same result as “hostname.exe”¹¹⁴. By the way, “hostname.exe” uses the Win32 API in order to retrieve the information, based on ReactOS¹¹⁵ the function is “GetComputerNameExW”¹¹⁶.

Moreover, for cases in which we have a cluster of compute nodes that have a distinct name we can set the environment variable “_CLUSTER_NETWORK_NAME_” which will change the data returned by Win32 API function¹¹⁷. Thus, the data returned by “hostname.exe” will also change as shown in the screenshot below.

Lastly, for an implementation reference of “hostname.exe” I suggest going over the one in ReacOS¹¹⁸.



```
C:\>hostname
DESKTOP-██████████

C:\>set _CLUSTER_NETWORK_NAME_=Troller

C:\>hostname
Troller
```

¹¹⁴ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/hostname>

¹¹⁵ <https://github.com/reactos/reactos/blob/3fa57b8ff7fcee47b8e2ed869aecaf4515603f3f/base/applications/cmdutils/hostname/hostname.c#L36>

¹¹⁶ <https://learn.microsoft.com/en-us/windows/win32/api/sysinfoapi/nf-sysinfoapi-getcomputernameexw>

¹¹⁷ <https://jeffpar.github.io/kbarchive/kb/198/Q198893/>

¹¹⁸ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aecaf4515603f3f/base/applications/cmdutils/hostname>

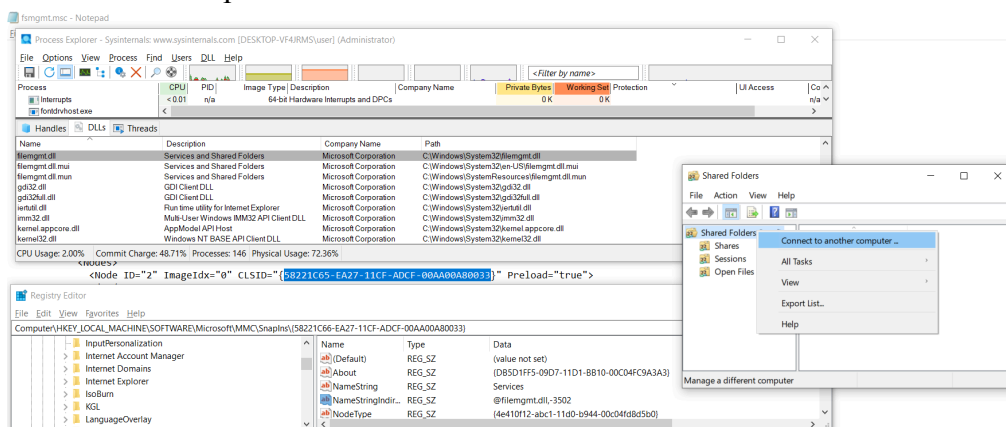
mmc.exe (Microsoft Management Console)

“mmc.exe” is the “Microsoft Management Console” which is responsible for creating/saving/opening consoles (aka administrative tools). They are used in order to manage software/hardware/network components as part of a given system which runs Windows. We can also create our own custom console and distribute it. Those consoles can include different snap-ins, which is a management tool hosted by “mmc.exe”¹¹⁹.

Moreover, snap-ins/custom console are distributed as part of “*.msc” file, which are as of today are XML files that are parsed “mmc.exe” in order to load the specific snap-ins¹²⁰. Even a clean installation of Windows comes with a couple of builtin “*.msc” file like: “services.msc” (for managing services), “WF.msc” (for managing the “Windows Defender Firewall”) and “fsmgmt.msc” (for managing shared folders). You can find them (and more) in the following location: “%windir%\system32\” (of course we can also save them to other locations).

At the end, a snap-in leads to a specific “*.dll” which is loaded by “mmc.exe” (“*.msc” can include a reference for a couple of snap-ins). The relevant configuration is stored in the registry under “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MMC\SnapIns”¹²¹. The snap-ins are identified using a “CLSID” (as other COM objects) - as seen in the screenshot below. Fun fact about “*.msc” files contain data of the icon we want to be displayed when the file is shown by “explorer.exe”¹²² or when “mmc.exe” is executed (as the app icon).

Also, one of the differences between MMC and other management consoles in Windows (like “Control Panel”) is the fact we can also manage remote systems (we have to authenticate for that) - as shown in the screenshot below (on the right side). Lastly, a reference implementation of “mmc.exe” is included as part of ReactOS¹²³.



¹¹⁹<https://learn.microsoft.com/en-us/troubleshoot/windows-server/system-management-components/what-is-microsoft-management-console>

¹²⁰http://file.fyicenter.com/143_Windows_MSC_File_Extension_for_Microsoft_Management_Conso.html

¹²¹<https://www.groovypost.com/tips/mmc-exe-windows-process-safe-virus/>

¹²²<https://medium.com/@boutnaru/the-windows-process-journey-explorer-exe-windows-explorer-9a96bc79e183>

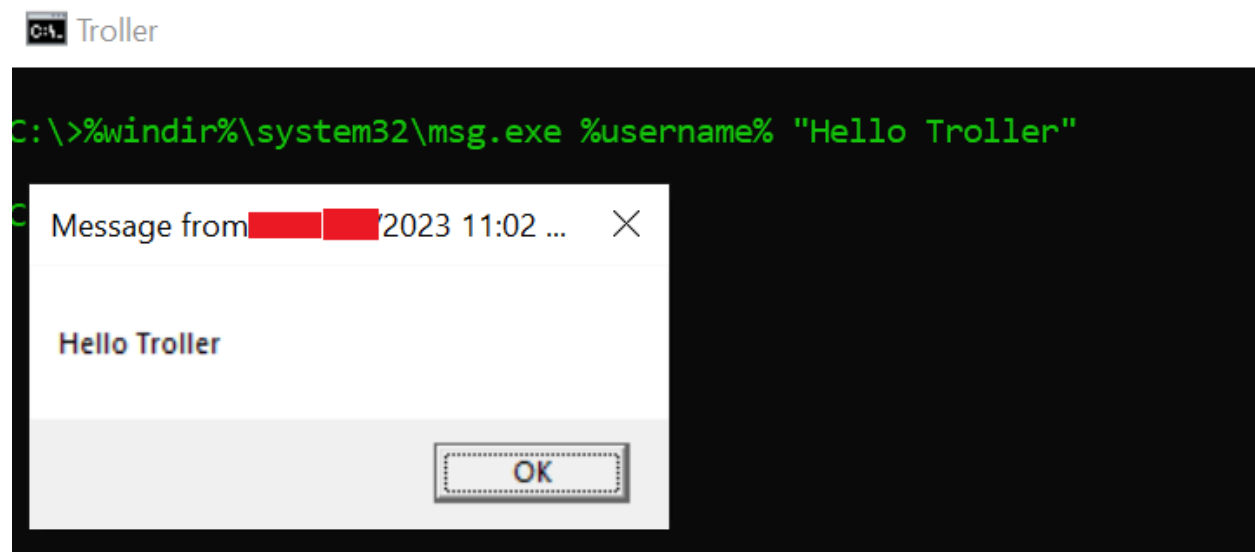
¹²³<https://github.com/reactos/reactos/tree/master/base/applications/mmc>

msg.exe (Message Utility)

“msg.exe” is the “Message Utility” which is a command line which allows sending a message to a user. It is a PE binary located at “%windir%\System32\msg.exe” which is signed by Microsoft. On a 64-bit system there is no 32-bit version of this file (in the SysWOW64 directory).

Overall, we can send a message by specifying a username (using * causes the message to arrive to all users), a session id and even send a message to a remote machine, it is mainly used for sending Terminal Services/Citrix shutdown messages. Also, we can define a delay for waiting for the receiver to acknowledge the message. The executable is not included in ‘Home’ editions of Windows¹²⁴.

Moreover, historically this functionality was part of the “Messenger Service” until Windows Vista/2008. It was also operated by using the “net send” command¹²⁵. Lastly, the sending of the message is done using RPC (“msg.exe” loads the RPC runtime DLL) and even MS-RPC over SMB in case of sending the message to a remote¹²⁶. We can see an example of using “msg.exe” in the screenshot shown below.



¹²⁴ <https://ss64.com/nt/msg.html>

¹²⁵ <https://www.lifewire.com/net-send-2618095>

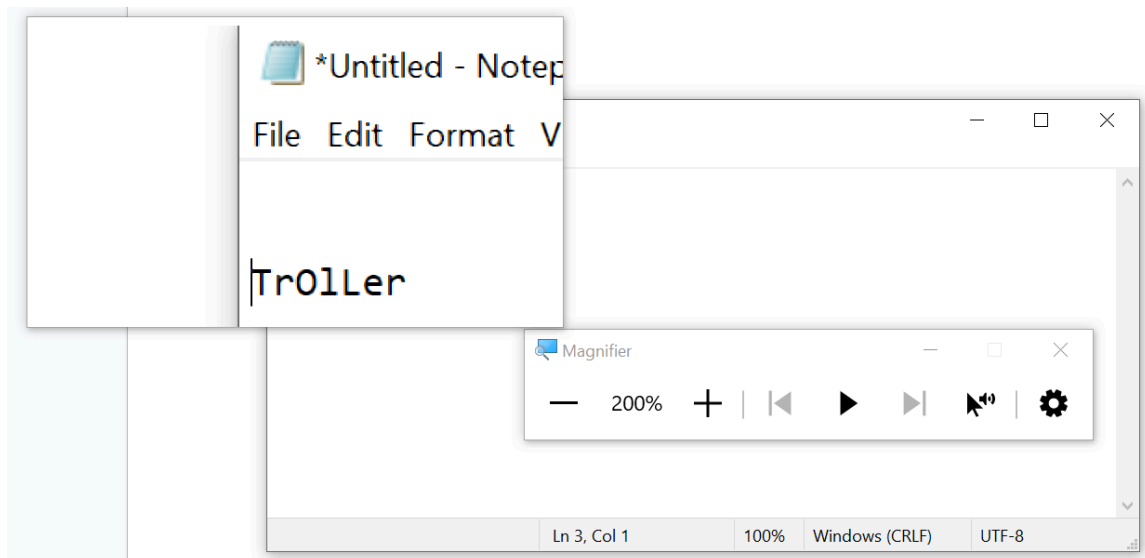
¹²⁶ <https://sid-500.com/2017/10/07/active-directory-send-messages-to-all-currently-logged-on-users-msg-exe/comment-page-1/>

Magnify.exe (Microsoft Screen Magnifier)

“Magnify.exe” is the “Microsoft Screen Magnifier” which makes part of the screen bigger in order to see images/text better. “Magnify.exe” has several options like: customizing the zoom level, smoothing the edges of images/text, inverting colors, reading text and more¹²⁷

Overall, “Magnify.exe” is a PE binary located at “%windir%\System32\Magnify.exe” which is signed by Microsoft. Also, on a 64-bit system there is also a 32-bit version located at “%windir%\SysWOW64\Magnify.exe”. Also, the file is signed by Microsoft.

Lastly, although there is no help displayed by “Magnify.exe” when running it from the command line it still has a couple of switches that can be used. Examples are “/lens” (as shown in the screenshot below) which defaults to lens view and “/docked” which defaults to “dock view”¹²⁸.



¹²⁷<https://support.microsoft.com/en-us/windows/use-magnifier-to-make-things-on-the-screen-easier-to-see-414948ba-8b1c-d3bd-8615-0e5e32204198>

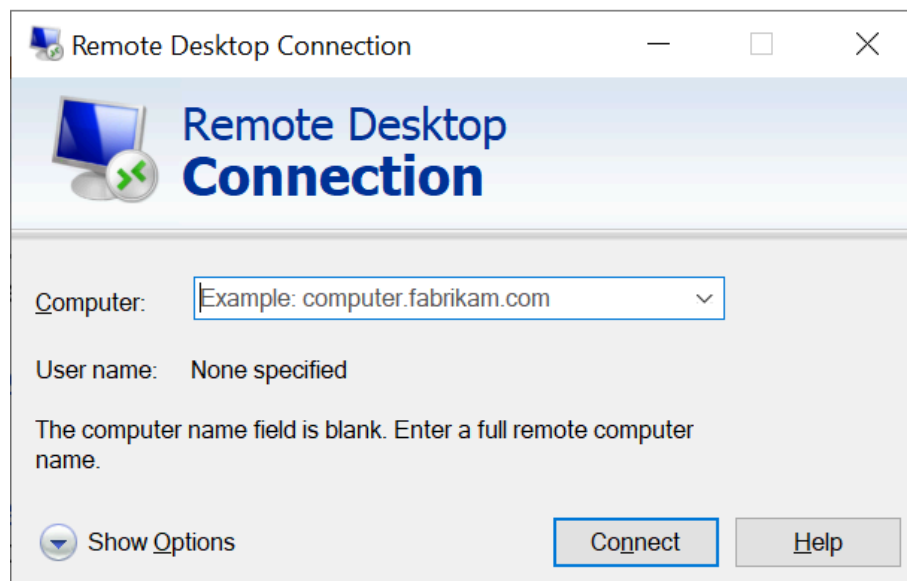
¹²⁸<https://answers.microsoft.com/en-us/windows/forum/all/magnifyexe-zoom-in-from-cmd-command-prompt/48c7257b-c1f8-483c-a0b8-ff24daf1622>

mstsc.exe (Remote Desktop Connection)

“mstsc.exe” is an executable located at “%windir%\System32\mstsc.exe”, it is also known as “Remote Desktop Connection”. On a 64-bit system there is also a 32-bit version located at “%windir%\SysWOW64\mstsc.exe”. It is a PE file which is signed by Microsoft.

Moreover, the name of the executable comes from “Microsoft Terminal Service Client”. “Terminal Service” was the previous name for the protocol used for the remote connection. Today it is called “Remote Desktop Protocol” (RDP). “mstsc.exe” is the default client for RDP that is part of the Windows operating system¹²⁹. I will write a dedicated writeup about the RDP protocol itself.

Overall, “mstsc.exe” allows users to connect to a “Remote Session Host” server or remote computer and to use the GUI interface of the remote system. Also, by using the executable we can edit “*.rdp” file, which is a remote desktop connection configuration file¹³⁰. Using “mstsc.exe” a user can also share its printers/clipboard/audio devices/network drives with the remote system to which the connection is being done. Lastly, for an implementation reference of “mstsc.exe” I suggest going over the one in ReacOS¹³¹.



¹²⁹ https://en.wikipedia.org/wiki/Remote_Desktop_Protocol

¹³⁰ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/mstsc>

¹³¹ <https://github.com/reactos/reactos/tree/3fa57b8ff7fcee47b8e2ed869aeca4515603f3f/base/applications/mstsc>

curl.exe (cURL executable)

“curl.exe” is a command line tool which allows transferring data with URLs. It supports various protocols like: FTP/S, HTTP/S, IMAP/S, LDAP/S, MQTT, POP3, SMB/S¹³². “curl” is a popular command line tool for Linux¹³³. There is also a version of “curl” for Windows. it is statically linked with different libraries like: libssh2, brotli, zlib, zstd, nghttp3, nghttp2, cacert¹³⁴.

Moreover, since build 17063 of Windows 10 (December 2017), Microsoft has announced that “curl” is going to be shipped by default as part of Windows¹³⁵. However, “curl.exe” that is shipped with Windows is handled and built by Microsoft. Microsoft’s version of “curl” uses the SChannel TLS backend¹³⁶.

Lastly, there is also a “curl” command as part of Powershell, but it is just an alias to the “Invoke-WebRequest”cmdlet - as shown in the screenshot below. We can go over the source code of curl in GitHub¹³⁷. Using “curl.exe” we can send HTTP GET requests (as shown below), resuming downloads, specifying max transfer rate and more¹³⁸.

```
Windows PowerShell
PS C:\> curl

cmdlet Invoke-WebRequest at command pipeline position 1
Supply values for the following parameters:
Uri:
PS C:\> C:\windows\system32\curl.exe
curl: try 'curl --help' for more information

C:\Windows\system32\cmd.exe

C:\> curl https://www.microsoft.com
<html><head><title>Microsoft Corporation</title><meta http-equiv="X-UA-Compatible" conte
nt="IE=EmulateIE7"></meta><meta http-equiv="Content-Type" content="text/html; charset=utf
-8"></meta><meta name="SearchTitle" content="Microsoft.com" scheme=""></meta><meta name
="Description" content="Get product information, support, and news from Microsoft." sche
me=""></meta><meta name="Title" content="Microsoft.com Home Page" scheme=""></meta><meta
name="Keywords" content="Microsoft, product, support, help, training, Office, Windows,
software, download, trial, preview, demo, business, security, update, free, computer, P
C, server, search, download, install, news" scheme=""></meta><meta name="SearchDescripti
on" content="Microsoft.com Homepage" scheme=""></meta></head><body><p>Your current User-
Agent string appears to be from an automated process, if this is incorrect, please click
this link:<a href="http://www.microsoft.com/en/us/default.aspx?redir=true">United State
s English Microsoft Homepage</a></p></body></html>

C:\>
```

¹³² <https://curl.se/>

¹³³ <https://linux.die.net/man/1/curl>

¹³⁴ <https://curl.se/windows/>

¹³⁵ <https://learn.microsoft.com/en-us/virtualization/community/team-blog/2017/20171219-tar-and-curl-come-to-windows>

¹³⁶ <https://curl.se/windows/microsoft.html>

¹³⁷ <https://github.com/curl/curl>

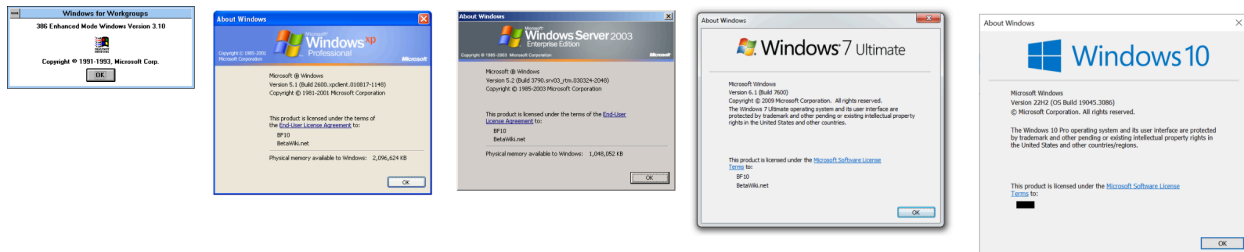
¹³⁸ <https://www.keycdn.com/support/popular-curl-examples>

winver.exe (Version Reporter Applet)

“winver” is the “Version Reporter Applet” which is responsible for displaying information about the version of the running operating system. It is also referred to as the “Windows Version” utility¹³⁹. It is a PE binary file located at “%windir%\System32\winver.exe”, on 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\winver.exe”.

Also, “winver.exe” is signed by Microsoft. It was first include in Windows from “Windows 3.0”, since “Windows 3.5” it calls the “ShellAbout” function from “shell32.dll”¹⁴⁰. Thus, if we have a version of Windows that does not include “winver.exe”(like Windows PE) we can use “rundll32.exe”¹⁴¹ to call it with the following command “rundll32 shell32,ShellAbout”.

Moreover, due to the UI changes that have been made in Windows along the way in Windows caused also for changes in “winver.exe” as shown in the screenshots below¹⁴². The examples are from the following versions of Windows (from left to right): “Windows 3.10”, “Window XP”, “Windows 2003 Server”, “Windows 7” and “Windows 10”. Lastly, we can checkout the implementation of “winver.exe” as part of ReacOS¹⁴³.



¹³⁹ <https://betawiki.net/wiki/Winver>

¹⁴⁰ <https://learn.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellaboutw>

¹⁴¹ <https://medium.com/@boutnaru/the-windows-process-journey-rundll32-exe-windows-host-process-415132f1363>

¹⁴² <https://betawiki.net/wiki/Winver>

¹⁴³ <https://github.com/reactos/reactos/tree/master/base/applications/winver>

arp.exe (TCP/IP Arp Command)

“arp.exe” (TCP/IP Arp Command) is a PE binary located at “%windir%\System32\ARP.EXE”. On 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\ARP.EXE”. Also, the binary file is digitally signed by Microsoft.

Overall, “arp.exe” allows displaying (using the “-a” or “/a” switch - as shown in the screenshot below) and modifying (using the “-s” or “/s” switch) entries in the ARP (Address Resolution Protocol) cache. There is a separate table for each network adapter that the system has (which is connected and has IP information). It is relevant for Ethernet/Token Ring network adapters¹⁴⁴.

Basically, ARP is a network protocol used for retrieving the link layer address (like MAC) for a given internet layer address (like IPv4). By the way, in IPv6 the functionality of ARP is implemented by NDP (Neighbor Discovery Protocol). Lastly, ARP is a request/response protocol which is encapsulated by the link layer protocol. Also, it is never routed across inter-networking entities¹⁴⁵.

```
C:\>arp -a

Interface: 192.168.157.180 --- 0x6
Internet Address      Physical Address      Type
192.168. [redacted]    00-15-5d-77-fc-00    dynamic
192.168. [redacted] 255    ff-ff-ff-ff-ff-ff    static
224.0.0.22           01- [redacted] -16    static
224.0.0.251         01- [redacted] -fb    static
224.0.0.252         01- [redacted] -fc    static
239.255.255.250     01- [redacted] -fa    static
255.255.255.255     ff-ff-ff-ff-ff-ff    static
```

¹⁴⁴ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/arp>

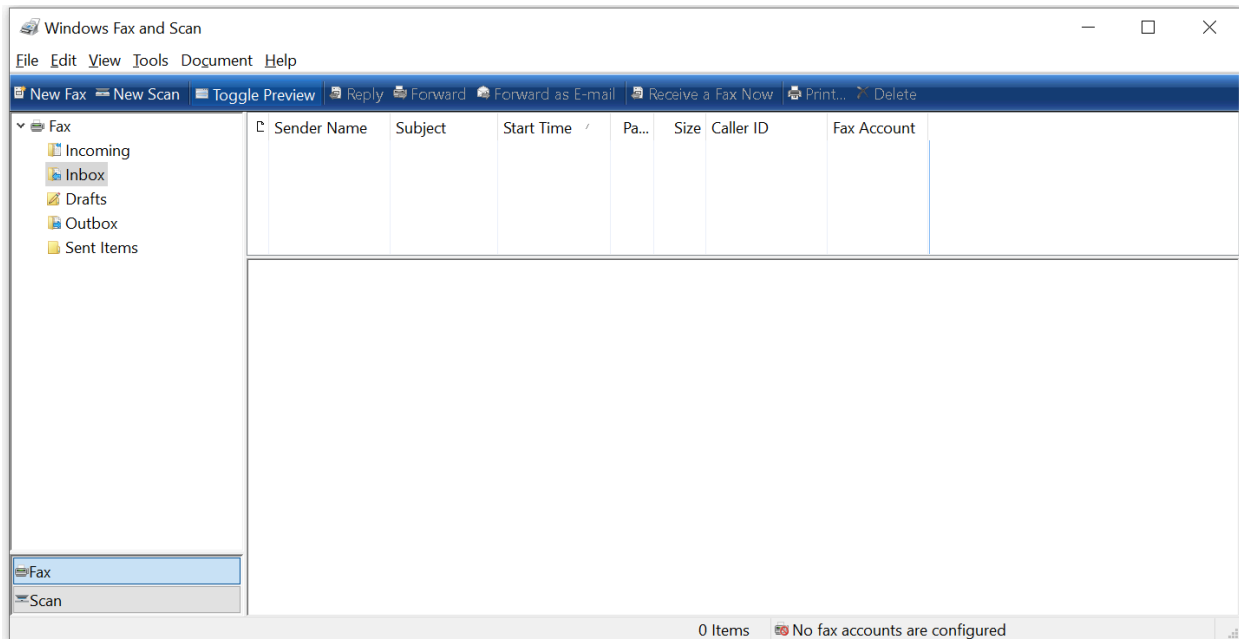
¹⁴⁵ https://en.wikipedia.org/wiki/Address_Resolution_Protocol

WFS.exe (Microsoft Windows Fax and Scan)

“WFS.exe” (aka the “Microsoft Windows Fax and Scan”) which is an integrated scanning and faxing app as part of Windows. It is the replacement of the “Fax Console” that was part of Windows XP. Overall, “WFS.exe” provides the ability to send/receive faxes, emailing/faxing scanned documents and forwarding faxes as email attachments¹⁴⁶.

Also, It is a PE binary file located at “%windir%\System32\WFS.exe” which is digitally signed by Microsoft. By the way, on a 64-bit system there is only the 64-bit version, there is not a 32-bit version (in “%windir%\SysWOW64”) like we have with other executables such as “cmd.exe”.

Moreover, in order for “WFS.exe” to operate correctly we need to install it as an “Optional Feature”¹⁴⁷. It is also dependent on the Fax service, which executable is located at “%windir%\System32\FXSSVC.exe”¹⁴⁸.



¹⁴⁶ https://en.wikipedia.org/wiki/Windows_Fax_and_Scan

¹⁴⁷ <https://www.intowindows.com/how-to-install-windows-fax-and-scan-in-windows-11/>

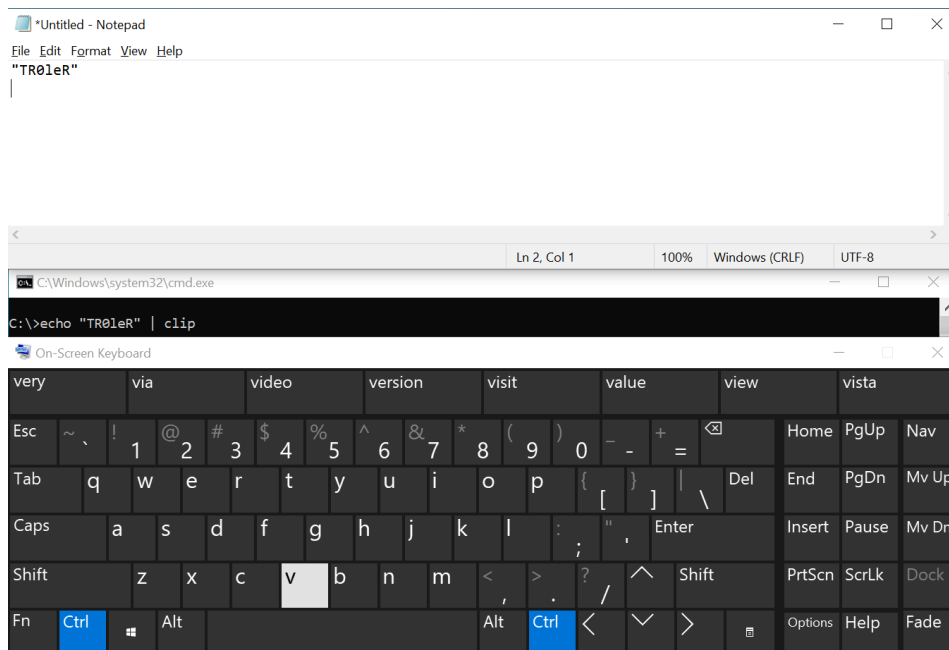
¹⁴⁸ [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc725953\(v=ws.11\)?re-directedfrom=MSDN](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2008-R2-and-2008/cc725953(v=ws.11)?re-directedfrom=MSDN)

clip.exe (Copies the Data into Clipboard)

“clip.exe” (copies the data into clipboard) is a PE binary located at “%windir%\System32\clip.exe”. On 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\clip.exe”. Also, the binary file is a CLI tool which is digitally signed by Microsoft.

Overall, “clip.exe” is used in order to copy the results of commands into the Windows clipboard. We can use it in one of the following ways: “command | clip” or “clip < file.txt”¹⁴⁹. After using “clip.exe” the text output can be pasted into another program.

Thus, we can see an example of usage in the screenshot below. In the screenshot we use “clip.exe” to store an echoed string into the clipboard. Using “osk.exe” (<https://medium.com/@boutnaru/the-windows-process-journey-osk-exe-accessibility-on-screen-keyboard-72823695321e>) aka the “On Screen Keyboard” we send “Ctrl+V” to paste the stored text into Notepad. Lastly, In powershell we have a cmdlet (“Set-Clipboard”) which does the same as “clip.exe” (<https://ss64.com/ps/set-clipboard.html>).



¹⁴⁹ <https://ss64.com/nt/clip.html>

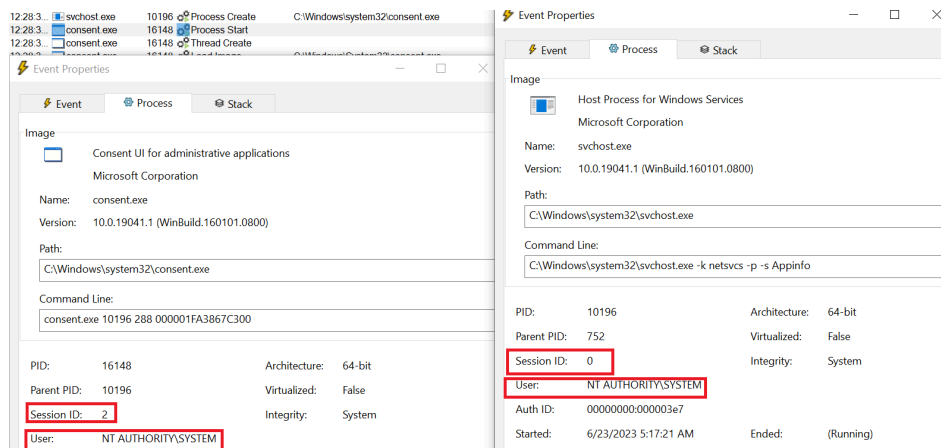
consent.exe (Consent UI for Administrative Applications)

“consent.exe” is the “Consent UI for Administrative Applications” which is called as part of a UAC (User Account Control) flow¹⁵⁰. It is a PE binary file located at “%windir%\system32\consent.exe”, which is signed digitally by Microsoft. On a 64-bit system there is no 32-bit version, as we have with other binaries such as “cmd.exe”.

Moreover, as shown in the screenshot below, “consent.exe” is started by the service “Application Information” which is hosted by “svchost.exe”¹⁵¹. The description of the service states that it “Facilitates the running of interactive applications with additional administrative privileges. If this service is stopped, users will be unable to launch applications with the additional administrative privileges they may require to perform desired user tasks”.

Also, as shown in the screenshot below, although it is running within “session 0” we can see that “consent.exe” is assigned to “session 2” with the permissions of “NT AUTHORITY\SYSTEM”. For further security the consent prompt is displayed on the secure desktop, only Windows processes can access the secure desktop¹⁵².

Lastly, if the logged on user is not an administrative account a credentials prompt will be displayed for getting a username and password for an administrative account - it is also done by “consent.exe” in a secure desktop¹⁵³. We can turn off prompting in secure mode with “reg.exe”:
‘REG ADD “HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System” /V “PromptOnSecureDesktop” /T “REG_DWORD” /D “0x00000000” /F’¹⁵⁴.



¹⁵⁰ <https://www.file.net/process/consent.exe.html>

¹⁵¹ <https://medium.com/@boutnaru/the-windows-process-journey-svchost-exe-host-process-for-windows-services-b18c65f7073f>

¹⁵² <https://learn.microsoft.com/en-us/windows/security/application-security/application-control/user-account-control/how-it-works>

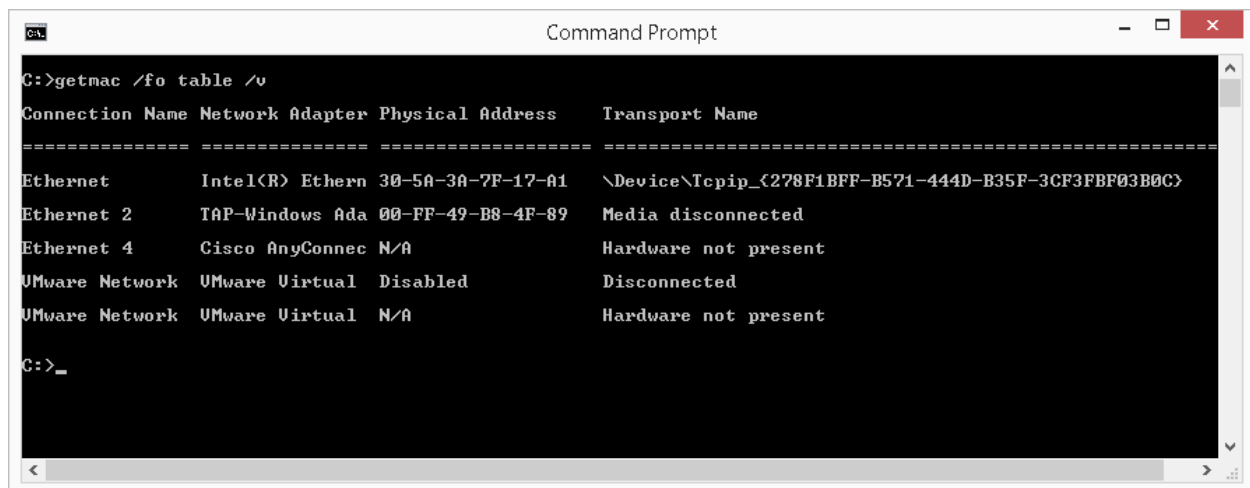
¹⁵³ <https://securityinternals.blogspot.com/2014/02/the-user-access-control-uac-prompts.html>

¹⁵⁴ <https://stackoverflow.com/questions/4046940/how-to-screen-shot-a-uac-prompt>

getmac.exe (Displays NIC MAC information)

“getmac.exe” is a binary PE file located at “%windir%\System32\getmac.exe”, on 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\getmac.exe”. This is a CLI application which is digitally signed by Microsoft.

Overall, “getmac.exe” is used for retrieving the MAC (Media Access Control) address for all the NIC (Network Interface Cards) on the system (both physical and virtual)¹⁵⁵. By the way, this is not the only CLI tool we can use to show the MAC address of NICs - we can also use “ipconfig.exe” (on which there is going to be a separate writeup) and even “nbtstat.exe” to show the MAC address of a remote machine (on this there is also going to be a separate writeup). Lastly, an example output of the command is shown below.



```
Command Prompt
C:>getmac /fo table /v
Connection Name Network Adapter Physical Address Transport Name
-----
Ethernet Intel(R) Ethern 30-5A-3A-7F-17-A1 \Device\Ncpip_{278F1BFF-B571-444D-B35F-3CF3FBF03B0C}
Ethernet 2 TAP-Windows Ada 00-FF-49-B8-4F-89 Media disconnected
Ethernet 4 Cisco AnyConnec N/A Hardware not present
VMware Network VMware Virtual Disabled Disconnected
VMware Network VMware Virtual N/A Hardware not present
C:>_
```

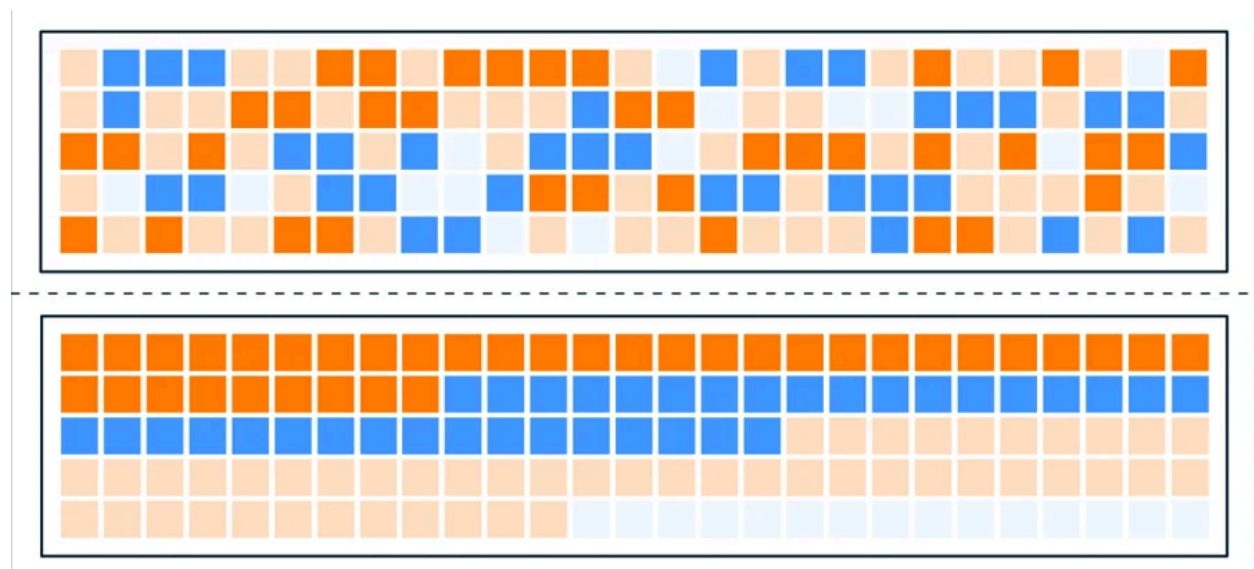
¹⁵⁵ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/getmac>

defrag.exe (Disk Defragmenter Module)

“defrag.exe” (Disk Defragmenter Module) is used to improve system performance by consolidating fragmented files on local volumes¹⁵⁶.

Overall, defragmentation organizes storage by consolidating files/other data saved on the hard drive. Due to different reasons when files are stored they can be broken down into smaller pieces (aka fragments) that can be spread across the hard drive. The goal of the defragmentation is to take scattered data in a hard drive and organize it for more efficient retrieval - as shown in the diagram below¹⁵⁷. The above part is before the process and the lower one is after it.

Moreover, we can't defragment every file system which exists. There is only support for NTFS, ReFS and FAT/FAT32 file system volumes. Thus, CD-ROMs/Network drives/volumes locked by the filesystem are not supported. Also, if the file system is marked as dirty, which might indicate possible corruption - it can be verified using the command “fsutil dirty”¹⁵⁸.



¹⁵⁶ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/defrag>

¹⁵⁷ <https://www.avast.com/c-how-to-defrag-pc-hard-drive>

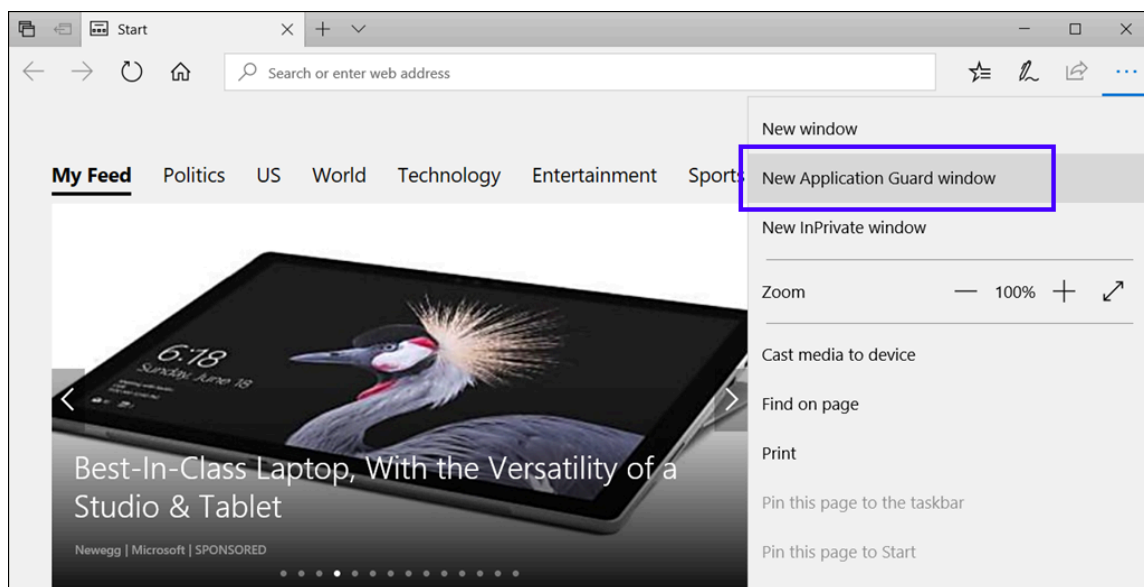
¹⁵⁸ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/defrag>

msedge.exe (Microsoft Edge)

“msedge.exe” is a 64-bit binary which is signed by Microsoft. Although it is a 64-bit binary it is still located by default in the program files directory of 32-bit applications ("C:\Program Files (x86)\Microsoft\Edge\Application\msedge.exe"). Microsoft Edge (aka Edge) is a web browser that is based on chromium which was released on January 15, 2020. It is supported on Windows, macOS, iOS and Android¹⁵⁹. By the way, if you want you can also be part of the “Microsoft Edge Insider Channel”. This allows you to be from the first who previews what’s new in Edge¹⁶⁰.

Moreover, from Windows 10 Enterprise/Pro (versions 1803 and later) or Windows 11 Pro users can use the “Application Guard” mode of Edge - as shown in the screenshot below. It disables printing from the application guard window, does not allow copying/pasting between the host PC and the application guard window and does not permit data persistence between application guard windows¹⁶¹.

Lastly, In order to enable that we need to enable the “Windows Defender Application Guard” feature (it requires the CPU support for virtualization). It launches Edge in an Hyper-V virtualized isolated environment¹⁶². A temporary container is created each time, it is destroyed/deleted when the user closes all the related windows¹⁶³.



¹⁵⁹<https://support.microsoft.com/en-us/microsoft-edge/download-the-new-microsoft-edge-based-on-chromium-0f4a3dd7-55df-60f5-739f-00010dba52cf>

¹⁶⁰<https://www.microsoft.com/en-us/edge/download/insider>

¹⁶¹<https://learn.microsoft.com/en-us/windows/security/application-security/application-isolation/microsoft-defender-application-guard/test-scenarios-md-app-guard>

¹⁶²<https://techcommunity.microsoft.com/t5/windows-insider-program/windows-defender-application-guard-standalone-mode/m-p/66903>

¹⁶³<https://blogs.windows.com/msedgedev/2016/09/27/application-guard-microsoft-edge/>

tzutil.exe (Windows Time Zone Utility)

“tzutil.exe” is a binary PE file located at “%windir%\system32\tzutil.exe”. It is used in order to display/set the time zone of the current system¹⁶⁴. On 64-bit systems there is also a 32-bit version of “tzutil.exe” located at “%windir%\SysWOW64\tzutil.exe”.

Moreover, “tzutil.exe” is a CLI tool which is digitally signed by Microsoft. For displaying the current time zone ID we use the “/g” switch while for setting the time zone we use the “/s” switch¹⁶⁵. There are different time zones that can be set using this command¹⁶⁶, we can also list them using the “/l” switch.

Lastly, there are cmdlets which are equal to “tzutil.exe” which is called Get-TimeZone/Set-TimeZone - as shown in the screenshot below.

```
C:\> C:\Windows\system32\cmd.exe
C:\>tzutil /g
Pacific Standard Time
C:\>powershell -c Get-TimeZone

Id                : Pacific Standard Time
DisplayName       : (UTC-08:00) Pacific Time (US & Canada)
StandardName     : Pacific Standard Time
DaylightName     : Pacific Daylight Time
BaseUtcOffset    : -08:00:00
SupportsDaylightSavingTime : True
```

¹⁶⁴ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/tzutil>

¹⁶⁵ <https://ss64.com/nt/tzutil.html>

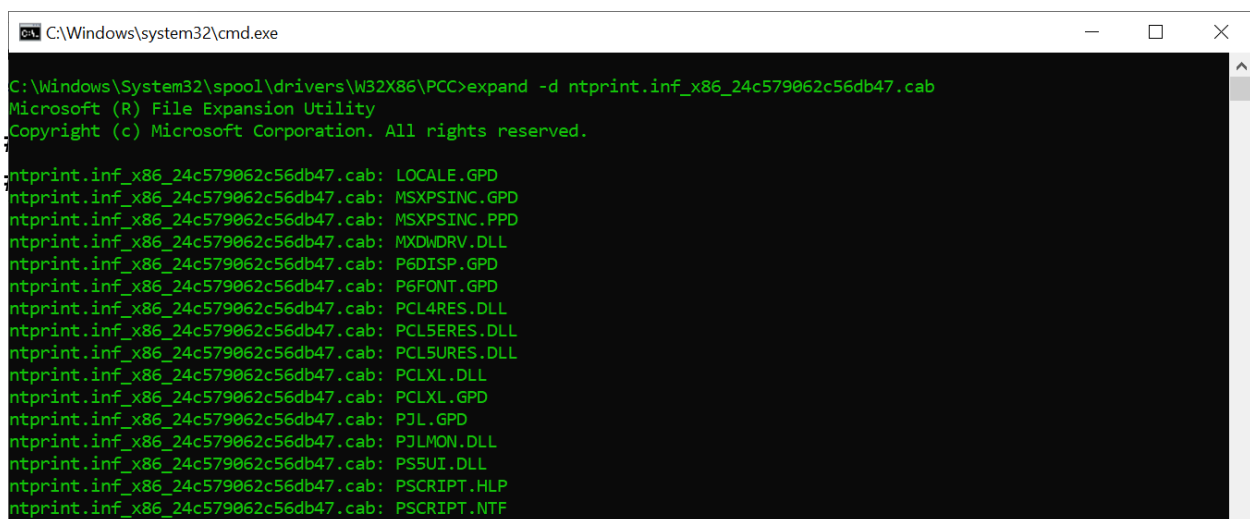
¹⁶⁶ <https://ss64.com/nt/timezones.html>

expand.exe (LZ Expansion Utility)

“expand.exe” aka “LZ Expansion Utility” is a PE binary located at “%windir%\System32\expand.exe”. It is used for expanding one or more compressed files. For example we can use it to retrieve compressed files from distribution disks¹⁶⁷. On 64-bit systems there is also a 32-bit version of “expand.exe” located at “%windir%\SysWOW64\expand.exe”.

Moreover, it is used to uncompress “*.cab” files (cabinet files). “expand.exe” is also called “The Microsoft File Expansion Utility” and it dates back to MS-DOS 5 in 1990¹⁶⁸. The simplest way to use it could be the following: “expand -d [FILE_NAME].cab” - as shown in the screenshot below.

Lastly, versions of expand before version 6.0 (Windows 7 timeline) included buggy implementation of “*.cab” file which include subfolders¹⁶⁹.



```
C:\Windows\system32\cmd.exe
C:\Windows\System32\spool\drivers\W32X86\PCC>expand -d ntprint.inf_x86_24c579062c56db47.cab
Microsoft (R) File Expansion Utility
Copyright (c) Microsoft Corporation. All rights reserved.

ntprint.inf_x86_24c579062c56db47.cab: LOCALE.GPD
ntprint.inf_x86_24c579062c56db47.cab: MSXPSINC.GPD
ntprint.inf_x86_24c579062c56db47.cab: MSXPSINC.PPD
ntprint.inf_x86_24c579062c56db47.cab: MXDWDREV.DLL
ntprint.inf_x86_24c579062c56db47.cab: P6DISP.GPD
ntprint.inf_x86_24c579062c56db47.cab: P6FONT.GPD
ntprint.inf_x86_24c579062c56db47.cab: PCL4RES.DLL
ntprint.inf_x86_24c579062c56db47.cab: PCL5ERES.DLL
ntprint.inf_x86_24c579062c56db47.cab: PCL5URES.DLL
ntprint.inf_x86_24c579062c56db47.cab: PCLXL.DLL
ntprint.inf_x86_24c579062c56db47.cab: PCLXL.GPD
ntprint.inf_x86_24c579062c56db47.cab: PJL.GPD
ntprint.inf_x86_24c579062c56db47.cab: PJLMON.DLL
ntprint.inf_x86_24c579062c56db47.cab: PSSUI.DLL
ntprint.inf_x86_24c579062c56db47.cab: PSCRIPT.HLP
ntprint.inf_x86_24c579062c56db47.cab: PSCRIPT.NTF
```

¹⁶⁷ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/expand>

¹⁶⁸ <https://ss64.com/nt/expand.html>

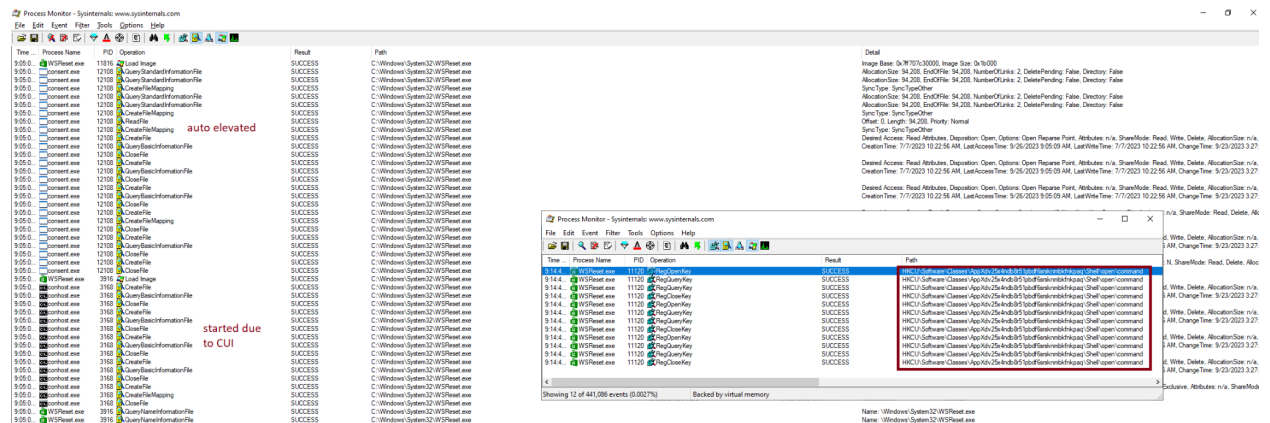
¹⁶⁹ <https://ss64.org/viewtopic.php?t=71>

WSReset.exe (Windows Store Reset)

In general, “WSReset.exe” is a PE binary file located at “%windir%\System32\WSReset.exe” which is also digitally signed by Microsoft. The description (Part of the PE format) states “This tool resets the Windows Store without changing account settings or deleting installed apps”. By the way, there is no 32-bit version of “WSReset.exe” on 64-bit systems (like we have with “cmd.exe” for example).

Thus, we can say “WSReset.exe” is used for clearing the cache of the “Windows Store”¹⁷⁰. The “Windows Store” creates temporary/cookies files in the following directories: “%UserProfile%\AppData\Local\Packages\Microsoft.WindowsStore_8wekyb3d8bbwe\AC\Net Cache” and “%UserProfile%\AppData\Local\Packages\Microsoft.WindowsStore_8wekyb3d8bbwe\AC\Net Cookies”. So in order to clear the cache of the executable just needs to delete the files from those folders¹⁷¹ - as also shown in the screenshot below.

Lastly, “WSReset.exe” is also auto elevated and during its startup it checks the following registry value “HKCU\Software\Classes\AppX82a6gwre4fdg3bt635tn5ctqjf8msdd2\Shell\open\command” for commands to execute¹⁷² - as shown in the screenshot below. This executable is a console tool, due to that “conhost.exe”¹⁷³ is also needed as we can see in the screenshot below.



¹⁷⁰ <https://helpdeskgEEK.com/how-to/how-to-clear-windows-store-cache-with-wsreset-exe/>

¹⁷¹ <https://daniels-it-blog.blogspot.com/2020/07/arbitrary-file-delete-via-wsresetexe.html>

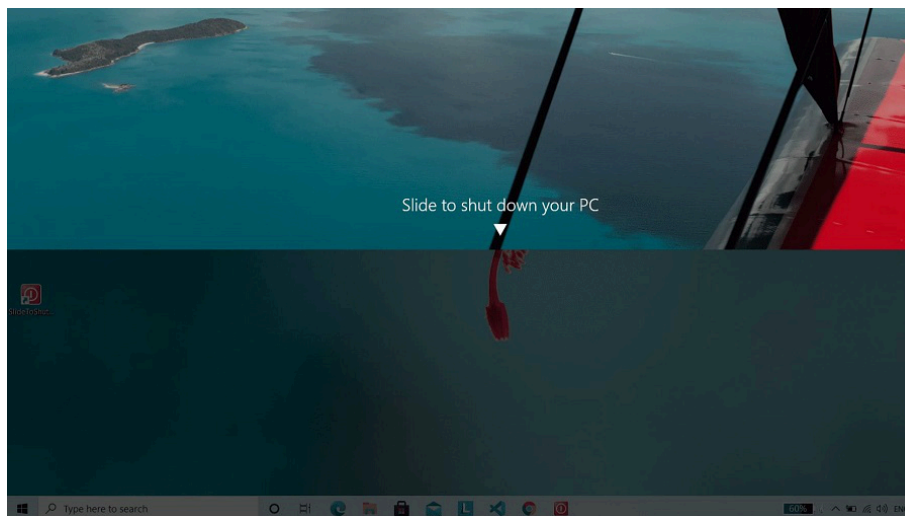
¹⁷² <https://lolbas-project.github.io/lolbas/Binaries/Wsreset/>

¹⁷³ <https://medium.com/@boutnaru/the-windows-process-journey-conhost-exe-console-window-host-f03f8db35574>

SlideToShutDown.exe (Windows Slide To Shutdown)

“SlideToShutDown.exe” is a PE binary located at “%windir%\System32\SlideToShutDown.exe”. It can be used in a smart and interactive way for shutting down Windows. Instead of the traditional way, we can just shutdown the system by sliding/dragging the window down - as shown in the screenshot below¹⁷⁴.

Moreover, on 64-bit systems we don't have a 32-bit version of “SlideToShutDown.exe” (as we have with “cmd.exe” for example). The binary is digitally signed by Microsoft. By default, the “slide to shutdown” should only show if we hold down the power button on a system with a touch screen¹⁷⁵. Lastly, even if we don't have a touch screen we can use the mouse for sliding/dragging the window down.



¹⁷⁴ <https://www.geeksforgeeks.org/creating-slide-to-shut-down-shortcut-in-windows-10/>

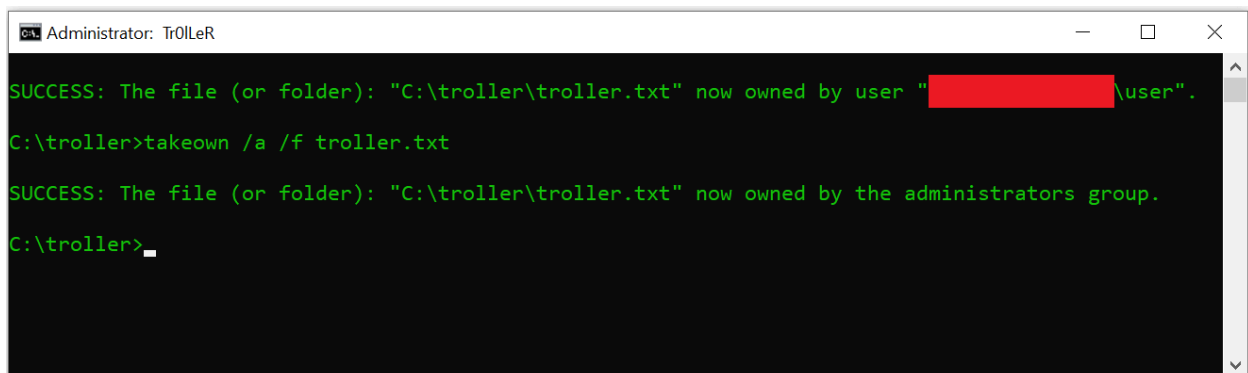
¹⁷⁵ <https://answers.microsoft.com/en-us/windows/forum/all/slide-to-shut-down/7b7e3f86-ccea-41a4-be8b-74531ea2fcb8>

takeown.exe (Takes Ownership of a File)

“takeown.exe” (Takes ownership of a file) is a PE binary located at “%windir%\System32\takeown.exe”. It is a CLI tool which allows an administrator to recover access to a file that was denied, it is done by changing the file-ownership¹⁷⁶. On 64-bit systems there is also a 32-bit version of “takeown.exe” located at “%windir%\SysWOW64\takeown.exe”.

Thus, after the ownership of the file/folder is taken the logged-on user is provided with the “full control” permissions. This allows the user to change the DACL¹⁷⁷ of the file/folder¹⁷⁸.

Lastly, by default the owner of a securable object¹⁷⁹ is based on the entity described by the access token¹⁸⁰ of the process/thread that has created it. It can be changed by the current owner or by a security context which holds the take ownership (SeTakeOwnershipPrivilege) privilege¹⁸¹.



```
Administrator: Tr0ll3r
SUCCESS: The file (or folder): "C:\troller\troller.txt" now owned by user "[redacted]\user".
C:\troller>takeown /a /f troller.txt
SUCCESS: The file (or folder): "C:\troller\troller.txt" now owned by the administrators group.
C:\troller>
```

¹⁷⁶ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/takeow>

¹⁷⁷ <https://medium.com/@boutnaru/the-windows-security-journey-dacl-discretionary-access-control-list-c74545e472ec>

¹⁷⁸ <https://appuals.com/takeown/>

¹⁷⁹ <https://medium.com/@boutnaru/windows-securable-objects-311a9d6c83ad>

¹⁸⁰ <https://medium.com/@boutnaru/windows-security-access-token-81cd0000c64>

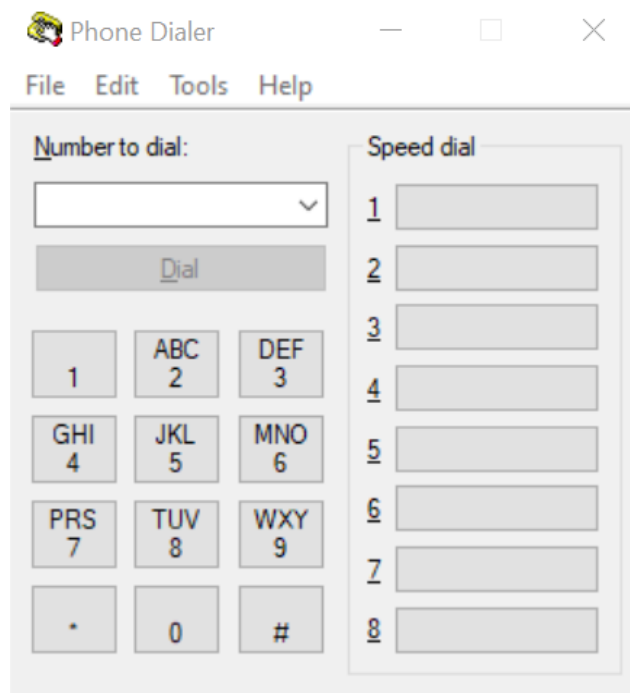
¹⁸¹ <https://medium.com/@boutnaru/windows-security-privileges-b8fe18cf3d5a>

dialer.exe (Microsoft Windows Phone Dialer)

“dialer.exe” (Microsoft Windows Phone Dialer) is a PE binary located at “%windir%\System32\dialer.exe”, which can be used to dial outgoing voice calls using the computer. It is done if the system has a modem supporting both voice and data¹⁸². On 64-bit systems there is also a 32-bit located at %windir%\SysWOW64\dialer.exe.

Thus, “dialer.exe” supports TAPI (Telephony Program Interface) based ActiveVoice¹⁸³. TAPI is an API (Application Programming Interface) allowing Windows systems to use the telephony services¹⁸⁴.

Moreover, TPAPI is a COM¹⁸⁵ based API that merges classic and IP telephony. It allows voice mailing, PBX control, basic voice over PSTN (Public Switched Telephone Network), call center applications, IVR (Interactive Voice Response), multicast multimedia and video conferencing¹⁸⁶. Lastly, we can think about “dialer.exe” as a software based phone - as also shown in the screenshot below.



¹⁸² <https://answers.microsoft.com/en-us/windows/forum/all/how-do-you-set-up-dialer/2aa4ef09-5a6d-4aa1-901b-557ff9ce0ef6>

¹⁸³ <https://answers.microsoft.com/en-us/windows/forum/all/dialerexe/b859ea03-f8f5-4b45-ab3a-19ff032763ff>

¹⁸⁴ https://documentation.avaya.com/en-US/bundle/IPOfficeSolutionDescription/page/Telephony_Application_Program_Interface.html

¹⁸⁵ <https://medium.com/@boutnaru/windows-com-component-object-model-71a76a97435c>

¹⁸⁶ <https://learn.microsoft.com/en-us/windows/win32/tapi/tapi-3-1-start-page>

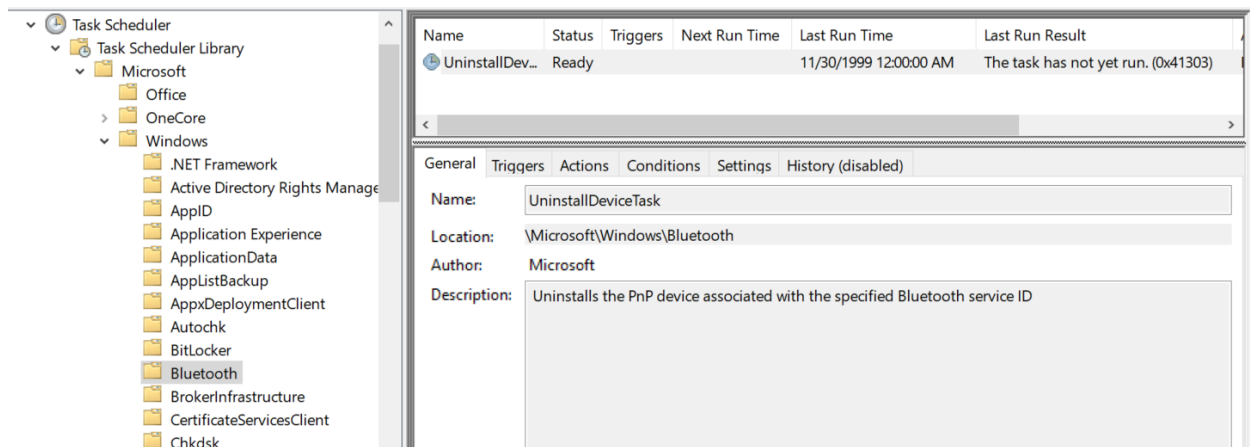
bthudtask.exe (Bluetooth Uninstall Device Task)

“bthudtask.exe” is a PE binary located at “%windir%\System32\bthudtask”, which is the Bluetooth uninstall device task. It is used to remove the pairing with a remote Bluetooth device, which is specified by service ID¹⁸⁷.

Moreover, on 64-bit systems there is also a 32-bit version of the executable located at “%windir%\SysWOW64\bthudtask.exe%”. Also, the executable is digitally signed by Microsoft and “auto elevated”.

Thus, the “Task Scheduler” task¹⁸⁸ that runs “bthudtask.exe” is “UninstallDeviceTask” which is located in the following hierarchy “Microsoft->Windows->Bluetooth” - as shown in the screenshot below. The scheduled task exits after the device is uninstalled¹⁸⁹.

Lastly, from the “Actions” tab we can see that the program is started “BthUdTask.exe \$(Arg0)”. This means that the Bluetooth service ID is given as the first argument.



¹⁸⁷ <https://www.shouldiblockit.com/bthudtask.exe-91.aspx>

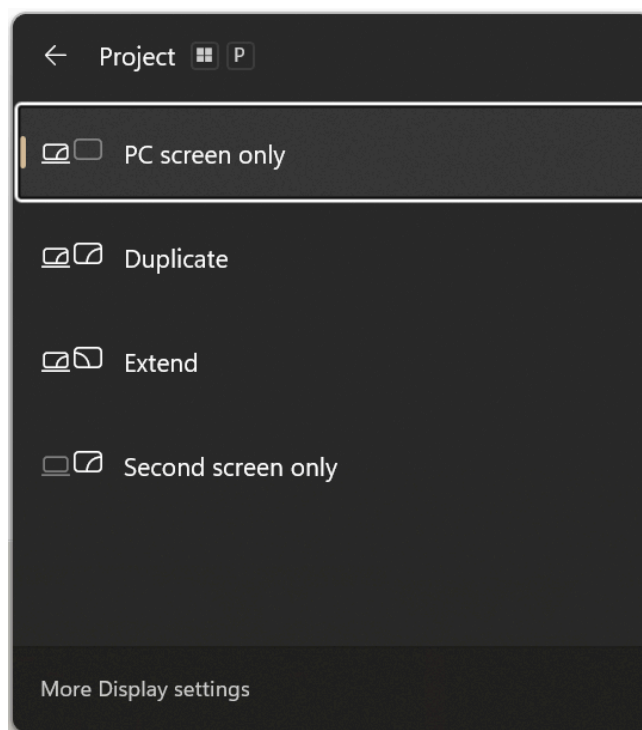
¹⁸⁸ <https://medium.com/@boutnaru/windows-scheduler-tasks-84d14fe733c0>

¹⁸⁹ <https://support.microsoft.com/en-gb/topic/description-of-the-scheduled-tasks-in-windows-vista-21f93b44-7260-a612-5ec3-fb2a7be5563c>

DisplaySwitch.exe (Windows Display Switch)

“DisplaySwitch.exe” is a PE binary located at “%windir%\System32\DisplaySwitch.exe”, it is used for switching the display based on different options like: PC only, duplicate (mirror), extend and second screen only - as shown in the screenshot below¹⁹⁰. Moreover, “DisplaySwitch.exe” is signed digitally by Microsoft. On a 64-bit system there is no 32-bit version of “DisplaySwitch.exe” (like we have for example with “cmd.exe”).

Lastly, on Windows 10 we can pass the following command line arguments: `/internal`, `/clone`, `/extend` and `/external` instead of selecting the option in the GUI. On Windows 11 the switches have been replaced with numbers: 1 (`=/internal`), 2 (`=/clone`), 3 (`=/extend`) and 4 (`=/external`). Keep in mind not to add a space after the number is given as input argument¹⁹¹.



¹⁹⁰ <https://learn.microsoft.com/en-us/answers/questions/1036148/displayswitch-exe-behavior-on-windows-11-22h2>

¹⁹¹ <https://learn.microsoft.com/en-us/answers/questions/1036148/displayswitch-exe-behavior-on-windows-11-22h2>

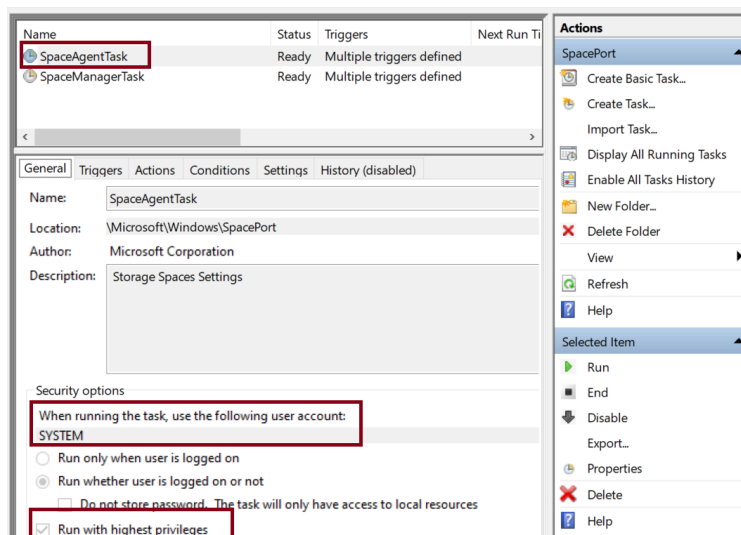
SpaceAgent.exe (Storage Spaces Settings)

“SpaceAgent.exe” is a PE binary located at “%windir%\System32\SpaceAgent.exe”. The description field in the PE format states it is “Storage Spaces Settings”. On 64-bit systems there is no 32-bit version of the binary - as we have with other binaries like “cmd.exe”¹⁹². It is good to know that the binary itself is also digitally signed by Microsoft.

Overall, “Storage Spaces” allows users to protect data from drive failures. It is a technology similar to RAID (Redundant Array of Independent Disks), which is implemented in software. “Storage Spaces” gives us the ability to combine three or more drives into a single pool of storage. This pool can then be used to create new storage spaces, which typically store multiple copies of your data for redundancy. So, if a drive fails, our data will still be safe¹⁹³.

Moreover, “SpaceAgent.exe” is configured to run as a scheduled task using the “Windows Scheduler”¹⁹⁴. We can see that configuration using the “Computer Management” console (“compmgmt.msc”) - as shown in the screenshot below. The task name is “SpaceAgentTask” and when executed it runs with the permissions of the “Local System” user - also shown in the screenshot. The location of the task configuration is in “%windir%\System32\Tasks\Microsoft\Windows\SpacePort\SpaceAgentTask”.

Lastly, from the manifest’s information as part of the “SpaceAgent.exe” binary, there is a description field which states: “Management agent for the Storage Spaces control panel applet”. Thus, if we click the “Storage Spaces” icon as part of the control panel and after that we click on “create new pool and storage spaces” an instance of “SpaceAgent.exe” is created.



¹⁹² <https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b>

¹⁹³ <https://learn.microsoft.com/en-us/windows-server/storage/storage-spaces/overview>

¹⁹⁴ <https://medium.com/@boutnaru/windows-scheduler-tasks-84d14fe733c0>

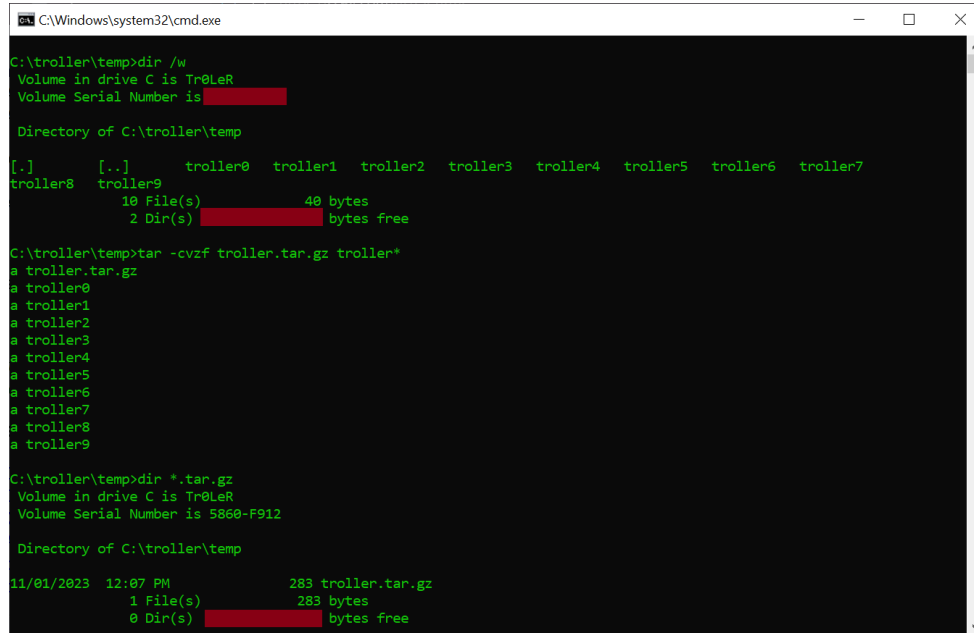
tar.exe (BSD tar Archive Tool)

“tar.exe” is a PE binary located at “%windir%\System32\tar.exe”. It is a command line tool which enables us to create archives and extract files¹⁹⁵. “tar.exe” is based on the “libarchive”¹⁹⁶, you can check out the code on GitHub¹⁹⁷. This is referenced by “tar.exe” by using “%windir%\System32\archiveint.dll”.

Moreover, “tar.exe” was added to Windows 10 (1803) from build 17063 or later as a pre-installed binary¹⁹⁸. There is also a 32-bit version of the binary located at “%windir%\SysWOW64\tar.exe”. Microsoft also digitally signs the “tar.exe” binary.

Overall, by going over the command line options of “tar.exe” we can see that we can perform different operations: create archives, list files inside archives, update archives and extract them. Also, we can compress an archive using gzip/bzip2/xz/lzma and use other formats ustar/pax/cpio/shar¹⁹⁹.

Lastly, when extracting an archive using “tar.exe” we can keep/overwrite existing files, restore (or not) modification times, write data to stdout (and not disk) and restore ACLs²⁰⁰ and other permission information (ownership and flags).



```
C:\Windows\system32\cmd.exe
C:\troller\temp>dir /w
Volume in drive C is Tr0LeR
Volume Serial Number is ██████████

Directory of C:\troller\temp

[.]          [..]          troller0  troller1  troller2  troller3  troller4  troller5  troller6  troller7
troller8     troller9
             10 File(s)      40 bytes
             2 Dir(s)      ██████████ bytes free

C:\troller\temp>tar -cvzf troller.tar.gz troller*
a troller.tar.gz
a troller0
a troller1
a troller2
a troller3
a troller4
a troller5
a troller6
a troller7
a troller8
a troller9

C:\troller\temp>dir *.tar.gz
Volume in drive C is Tr0LeR
Volume Serial Number is 5860-F912

Directory of C:\troller\temp

11/01/2023  12:07 PM                283 troller.tar.gz
             1 File(s)            283 bytes
             0 Dir(s)      ██████████ bytes free
```

¹⁹⁵ <https://learn.microsoft.com/en-us/virtualization/community/team-blog/2017/20171219-tar-and-curl-come-to-windows>

¹⁹⁶ <https://libarchive.org/>

¹⁹⁷ <https://github.com/libarchive/libarchive>

¹⁹⁸ https://renenyffenegger.ch/notes/Windows/dirs/Windows/System32/tar_exe

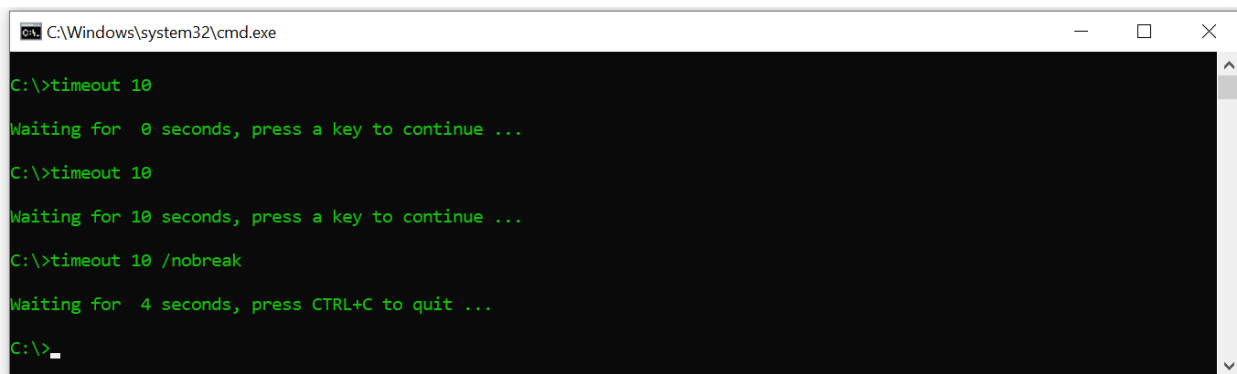
¹⁹⁹ <https://ss64.com/nt/tar.html>

²⁰⁰ <https://medium.com/@boutnaru/the-windows-security-journey-acl-access-control-list-b7d9a6fe428>

timeout.exe (Pauses Command Processing)

“timeout.exe” is a PE binary located at “%windir%\System32\timeout.exe”. It is a command line tool which enables pausing command processing. By using it we can delay execution for seconds/minutes as part of a batch file²⁰¹. By the way, we don’t have “sleep.exe” pre-installed on Windows, it is part of the “Windows Resource Kit”²⁰².

Moreover, on 64-bit systems of Windows we also have a 32-bit version of “timeout.exe” located at “%windir%\System32\timeout.exe”. It is also digitally signed by Microsoft. We can specify using a decimal number the amount of seconds we want to wait. The range is between (-1) to 99999. Using (-1) states to wait indefinitely for a key storkey. There is also an option of ignoring keystores using “/nobreak”, which can be canceled using “Ctrl+C”²⁰³. Lastly, we can see a couple of examples for using “timeout.exe” in the screenshot below.



```
C:\Windows\system32\cmd.exe
C:\>timeout 10
Waiting for 0 seconds, press a key to continue ...
C:\>timeout 10
Waiting for 10 seconds, press a key to continue ...
C:\>timeout 10 /nobreak
Waiting for 4 seconds, press CTRL+C to quit ...
C:\>_
```

²⁰¹ <https://ss64.com/nt/timeout.html>

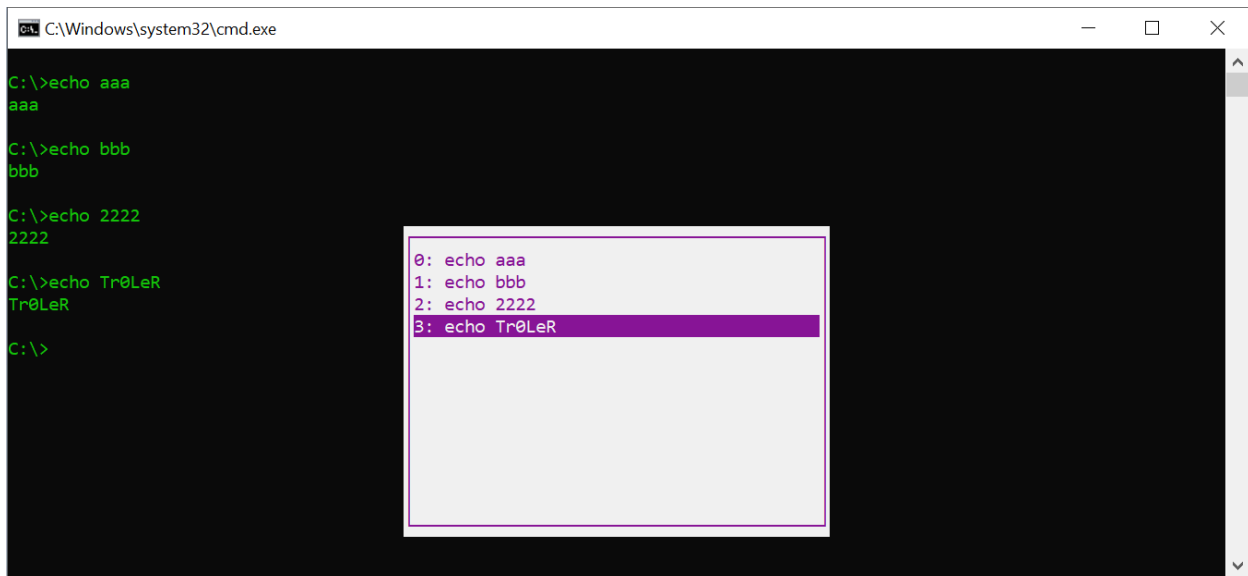
²⁰² <https://ss64.com/nt/sleep.html>

²⁰³ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/timeout>

doskey.exe (Keyboard History Utility)

“doskey.exe” (Keyboard History Utility) is a binary PE file located at “%windir%\system32\doskey.exe”. It is a CLI (command line interface) utility which is used for recalling previously entered commands. Also, we can use it for editing commands and creating macros²⁰⁴.

Moreover, after running “doskey.exe” we can use F7 in order to see the buffer/log/history of commands entered in a menu - as shown in the screenshot below. There are multiple keys/combinations that “doskey.exe” recognizes like “ALT+F7” which clears the history buffer and “End” which moves to the end of the line²⁰⁵. Lastly, we can go over a reference implementation of “doskey.exe” from ReactOS²⁰⁶.



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The prompt is at "C:\>". The user has entered four commands: "echo aaa", "echo bbb", "echo 2222", and "echo Tr0LeR". The output of these commands is displayed in green text. A doskey utility window is overlaid on the command prompt, showing a list of commands in a history buffer. The list is as follows:

```
0: echo aaa
1: echo bbb
2: echo 2222
3: echo Tr0LeR
```

The third item, "3: echo Tr0LeR", is highlighted with a purple background. The doskey window has a white background and a purple border.

²⁰⁴ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/doskey>

²⁰⁵ <https://kb.iu.edu/d/aers>

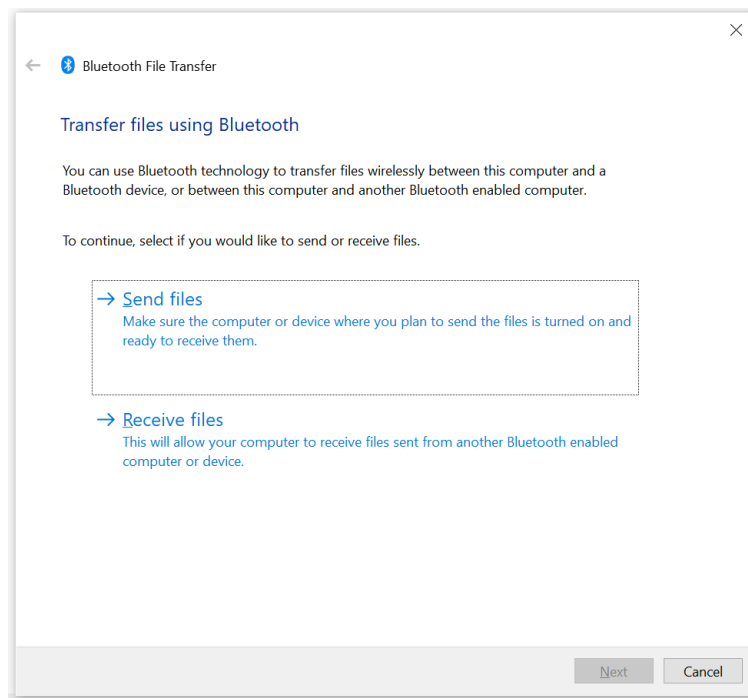
²⁰⁶ <https://github.com/reactos/reactos/tree/master/base/applications/cmdutils/doskey>

fsquirt.exe (Bluetooth File Transfer)

“fsquirt.exe” is a PE binary located at “%windir%\System32\fsquirt.exe” which is used for sending/receiving files using Bluetooth. On 64-bit systems there is a 32-bit version located at “%windir%\SysWOW64\fsquirt.exe”. By the way, the binary is also digitally signed by Microsoft.

Thus, “fsquirt.exe” is the default Bluetooth file transfer wizard on Windows systems²⁰⁷. The file transfer can be done between two computer that support Bluetooth, mobile phones or any other Bluetooth enabled devices²⁰⁸.

Lastly, “fsquirt.exe” is also configured in the registry in the following registry location: “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths”. The “App Paths” subkey is checked when the ShellExecuteExW²⁰⁹ API function is called (The same goes for ShellExecuteExA). By registering an application using that subkey we can avoid the need for modifying the PATH environment variable²¹⁰.



²⁰⁷ https://renenyffenegger.ch/notes/Windows/dirs/Windows/System32/fsquirt_exe

²⁰⁸ <https://learn.microsoft.com/en-us/windows-hardware/drivers/bluetooth/bluetooth-user-interface>

²⁰⁹ <https://learn.microsoft.com/en-us/windows/win32/api/shellapi/nf-shellapi-shellexecuteexW>

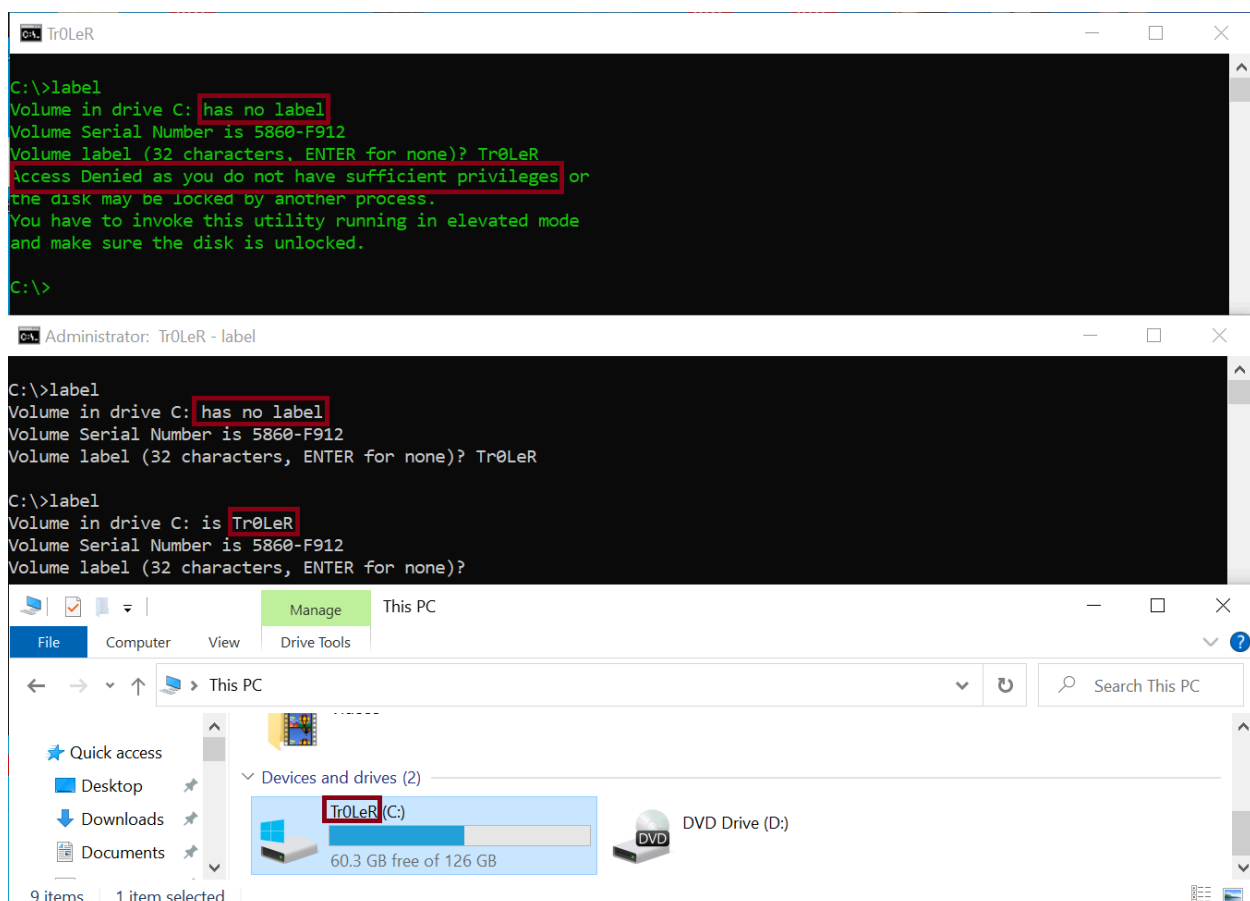
²¹⁰ <https://learn.microsoft.com/en-us/windows/win32/shell/app-registration>

label.exe (Disk Label Utility)

“label.exe” (Disk Label Utility) is a binary PE file located at “%windir%\system32\label.exe”. It is a CLI (command line interface) utility which is used for creating/changing/deleting the volume label of a disk²¹¹.

Moreover, on an NTFS volume we can use a label with up to 32 characters. On 64-bit systems there is also a 32-bit version on “label.exe” located at “%windir%\SysWOW64\label.exe”. Both versions of the PE are signed digitally by Microsoft.

The volume label is displayed in different places like in the “File Explorer” or the output of the “label.exe” - as marked in the screenshot below. In order to change the label there is a need for admin privileges - as shown in the screenshot below. Lastly, we can also go over a reference implementation of “label.exe” as part of ReactOS²¹².



²¹¹ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/label>

²¹² <https://github.com/reactos/reactos/tree/master/base/applications/cmdutils/label>

forfiles.exe (Execute a Command on Selected Files)

“forfiles.exe” is a binary PE file located at “%windir%\system32\forfiles.exe”. It is a CLI (command line interface) utility which can be used in order to execute a command on selected files. On 64-bit versions of Windows there is also a 32-bit version of the binary located at “%windir%\SysWOW64\forfiles.exe”. Also, the file is digitally signed by Microsoft.

Overall, “forfiles.exe” was included as part of Windows 98²¹³ and Windows 2000²¹⁴ resource kits, that means it was not part of the standard OS installation. Since Windows Vista it is part of the executables shipped with the OS²¹⁵.

Moreover, “forfiles.exe” has multiple command line parameters including: “/S” (recursive search), “/P” (specifying start directory), “/M” (search pattern mask), “/D” (selecting files by a last modification time frame), “/?” (displaying help text) and “/C” (specifying what command to run on each file). When using “/C” we can also use specific variables as part of the command like “@file” (the file name we are operating on), “@path” (the full path), “@ext” (the file extension) and more²¹⁶.

Lastly, we can see an example of using “forfiles.exe” in the screenshot below. In the screenshot we that for every file in the “C:\troller” directory with a “troller*” pattern in the file name we execute the type builtin command of “cmd.exe”²¹⁷.



```
C:\troller>forfiles /M troller* -c "cmd /c type @file"

tR0LLeR
TRoLleR
TrØLLeR
TrøLeR

C:\troller>
```

²¹³ <https://web.archive.org/web/20200111203651/https://www.activexperts.com/admin/reskit/reskit98/forfiles/>

²¹⁴ <https://www.activexperts.com/admin/reskit/reskit2000/forfiles/>

²¹⁵ <https://web.archive.org/web/20061109021306/http://computerbits.wordpress.com/2006/07/21/new-command-line-tools-in-vista-beta-2/>

²¹⁶ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/forfiles>

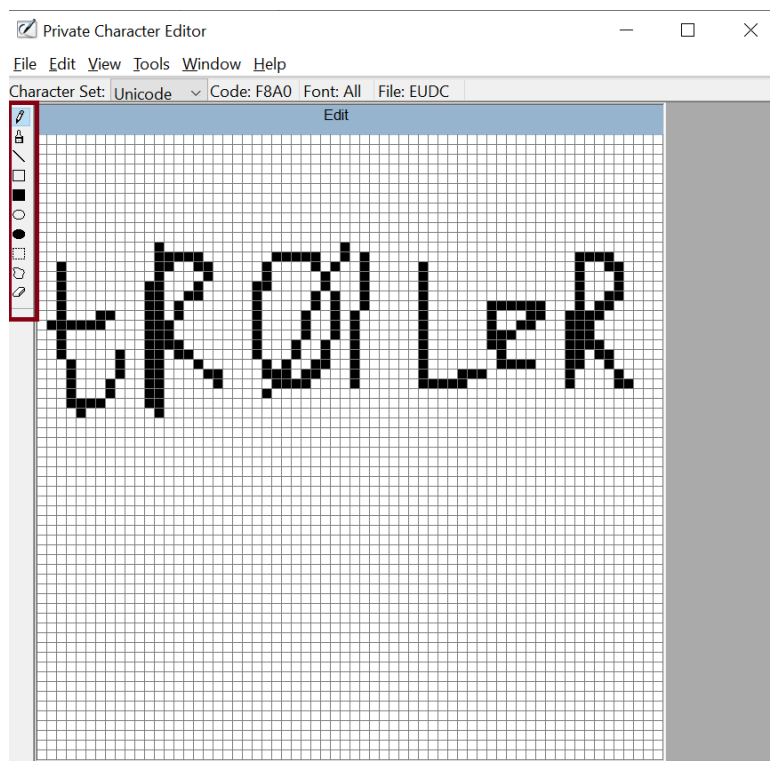
²¹⁷ <https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b>

eudcedit.exe (Private Character Editor)

“eudcedit.exe” is a PE binary located at “%windir%\System32\eudcedit.exe” it is known as the “Private Character Editor”. In case we want to use our own character/symbol (like in a document) we can use “eudcedit.exe”. Overall, it provides different tools for creating symbols/characters including: pencil, brush, eraser, hollow/filled eclipse/rectangles, straight line and rectangular/freeform selection²¹⁸.

Overall, we can create a character/symbol in one of two ways. First, creating a new custom one or second creating a custom one using a pre-existing character/symbol. By the way, on 64-bit versions of Windows there is also a 32-bit version of the binary located at “%windir%\SysWOW64\eudcedit.exe”. The binary itself is also digitally signed by Microsoft.

Lastly, “eudcedit.exe” is configured to be auto elevated by default (based on the manifest information included in the binary itself “<autoElevate>true</autoElevate>”). In the screenshot below we can see an example of using the editor and all the mentioned tools marked in the left side of the UI.



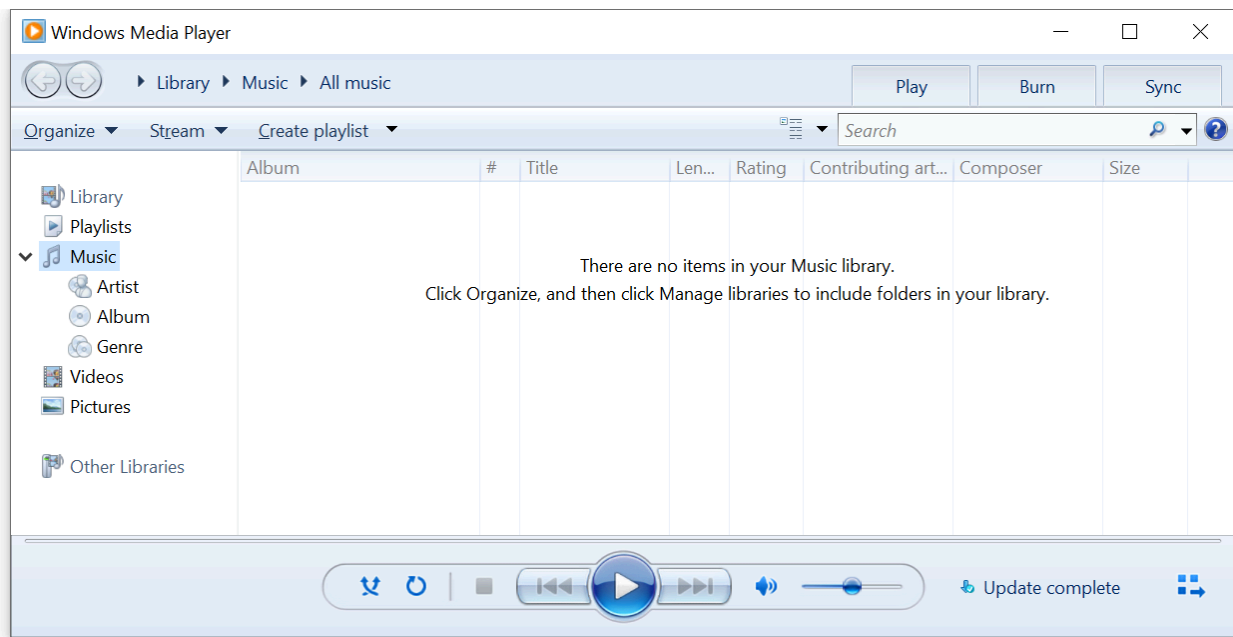
²¹⁸ <https://www.thewindowsclub.com/charmap-and-eudcedit-windows-10>

wmplayer.exe (Windows Media Player)

“wmplayer.exe” is a PE binary located at “%ProgramFiles(x86)%\Windows Media Player\wmplayer.exe”. It is used for It as a media player, which is an application used for playing multimedia files (video and audio). It can also be used as a media library application - as shown in the screenshot below. By the way, WMP (Windows Media Player) has been included since Windows 3.x²¹⁹. However, since 2022 it is marked as legacy while there is a new UWP based Media Player introduced in Windows 11²²⁰.

Moreover, we can find the new version in the Windows Store. This version is relevant for Windows 10 (19042.0 or higher) on Mobile/PC/HoloLens/Xbox console/Surface Hub targeting x86/x64/Arm64 architectures²²¹.

Overall, the “wmplayer.exe” which is executed by default is the 32-bit version of WMP. There could also be a 64-bit version in the following location: “%ProgramFiles%\Windows Media Player\wmplayer.exe”. By the way, both versions are digitally signed by Microsoft.



²¹⁹ <https://www.youtube.com/watch?v=imAUwksUIY>

²²⁰ https://en.wikipedia.org/wiki/Windows_Media_Player

²²¹ <https://apps.microsoft.com/detail/9WZDNCRFJ3PT>

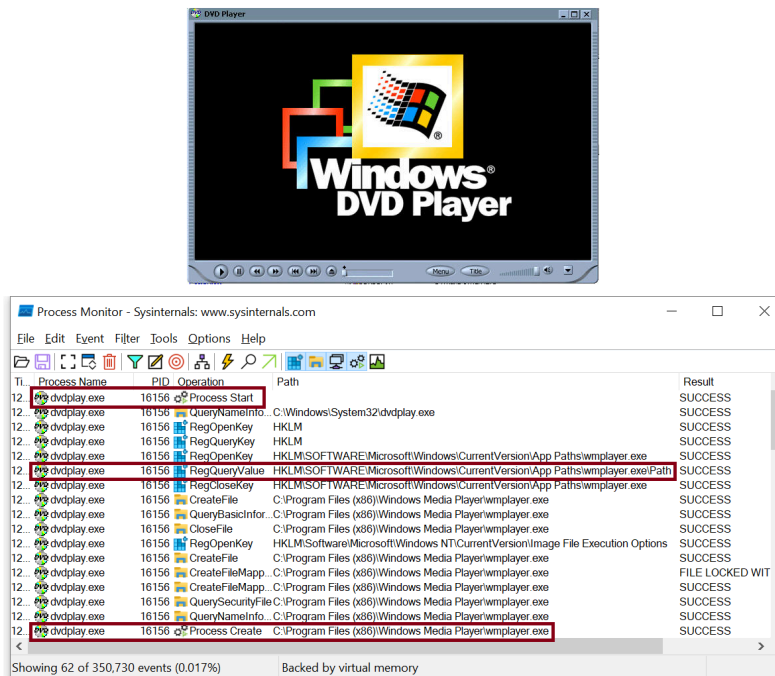
dvdplay.exe (DVD Play Placeholder Application)

“dvdplay.exe” is a PE binary located at “%windir%\System32\dvdplay.exe”. It is used for launching an application which is capable of playing DVD disks. On 64-bit versions of Windows there is also a 32-bit version of the binary located at “%windir%\SysWOW64\dvdplay.exe”. The binary is also digitally signed by Microsoft.

On old versions of Windows (like Windows ME), “dvdplay.exe” was its own application - as shown in the screenshot below²²². However, in new versions (like Windows 10) it is basically launching “wmplayer.exe” which is the “Windows Media Player”²²³.

Thus, “dvdplayer.exe” calls the API function “RegGetValueW”²²⁴ in order to read the path of “wmplayer.exe” from the application registration in the registry “HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\App Paths\wmplayer.exe\Path”. Later, it checks if the file exists using the API call “SearchPathW”²²⁵. If the file is found it is started using the API call “CreateProcessW”²²⁶.

Lastly, the flow described above aligns with the description found in the PE header which states it is a palace holder application. This flow is also shown in the screenshot below taken from Sysinternals’ “Process Monitor” on Windows 10.



²²² www.activewin.com/tips/tips/microsoft/winme/b3.shtml

²²³ <https://medium.com/@boutnaru/the-windows-process-journey-wmplayer-exe-windows-media-player-7d25c370c526>

²²⁴ <https://learn.microsoft.com/en-us/windows/win32/api/winreg/nf-winreg-reggetvaluew>

²²⁵ <https://learn.microsoft.com/en-us/windows/win32/api/processenv/nf-processenv-searchpathw>

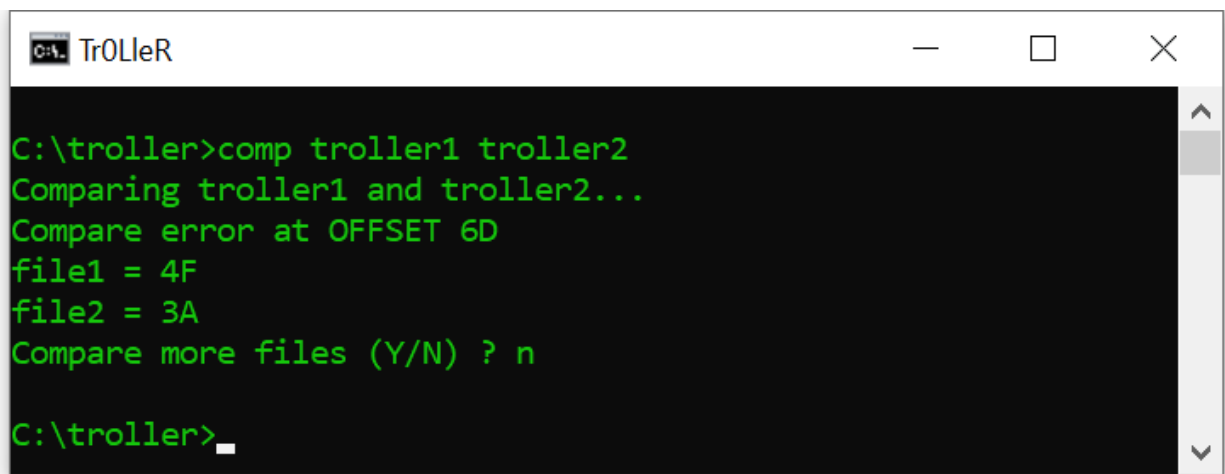
²²⁶ <https://learn.microsoft.com/en-us/windows/win32/api/processhthreadsapi/nf-processhthreadsapi-createprocessw>

comp.exe (File Compare Utility)

“comp.exe” is a PE binary located at “%windir%\System32\comp.exe”. It is used for comparing the content of two files/set of files byte-by-byte. The files compared may be located on the same drive/directory or on different drive/directory. On 64-bit systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\comp.exe”²²⁷.

Moreover, the files which are compared can also be in a remote location (SMB share). In case there is a difference between the compared files the offsets of change with the different values are displayed - as shown in the screenshot below. By the way, the “comp.exe” binary is also digitally signed by Microsoft.

Lastly, by using command line arguments we can display the difference in decimal (hex is the default), compare only a specific number of lines, display the difference in ascii characters and more²²⁸. Also, there is a reference implementation of “comp.exe” as part of ReactOS²²⁹.



```
C:\troller>comp troller1 troller2
Comparing troller1 and troller2...
Compare error at OFFSET 6D
file1 = 4F
file2 = 3A
Compare more files (Y/N) ? n
C:\troller>
```

²²⁷ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/comp>

²²⁸ <https://ss64.com/nt/comp.html>

²²⁹ <https://github.com/reactos/reactos/tree/master/base/applications/cmdutils/comp>

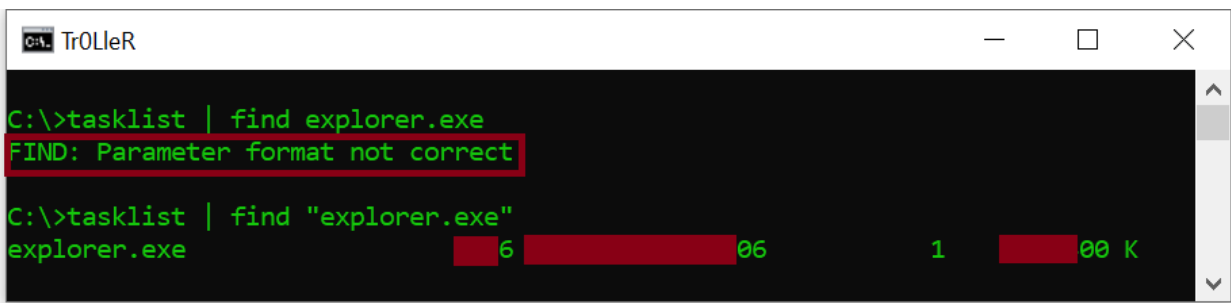
find.exe (Find String (grep) Utility)

“find.exe” is a PE binary located at “%windir%\System32\find.exe”. On 64-bit systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\find.exe”. Both of the versions are digitally signed by Microsoft. It is used in order to search for patterns of text files and sends them to the standard input device. Thus, we can use it to filter/find a specific string using wildcard characters²³⁰.

Overall, we can compare the functionality of “find.exe” to those of the “grep” utility²³¹ which is widely used under Unix/Linux systems. On the other hand it is completely different from the “find”²³² utility used in Unix/Linux systems which is similar to the “forfiles.exe”²³³.

Moreover, “find.exe” has different command line switches for: displaying all lines not containing a specific string (“/V”), counting the number of lines containing a string (“/C”), displaying line numbers (“/N”) and ignoring the case of characters while searching (“/I”). Also, we can skip (or not) files that have the offline attribute set²³⁴.

Lastly, we can provide a path/s to file/s (including wildcards) we want to search in their content, pass a standard output of a command as input or just get the input for a prompt by “find.exe”. It is important to understand that the string we want to search for must be in quotes - as shown in the screenshot below.



```
C:\>tasklist | find explorer.exe
FIND: Parameter format not correct

C:\>tasklist | find "explorer.exe"
explorer.exe           6          06          1          00 K
```

²³⁰ [https://en.wikipedia.org/wiki/Find_\(Windows\)](https://en.wikipedia.org/wiki/Find_(Windows))

²³¹ <https://man7.org/linux/man-pages/man1/grep.1.html>

²³² <https://man7.org/linux/man-pages/man1/find.1.html>

²³³ <https://medium.com/@boutnaru/the-windows-process-journey-forfiles-exe-execute-a-command-on-selected-files-3c10a9b2b5cf>

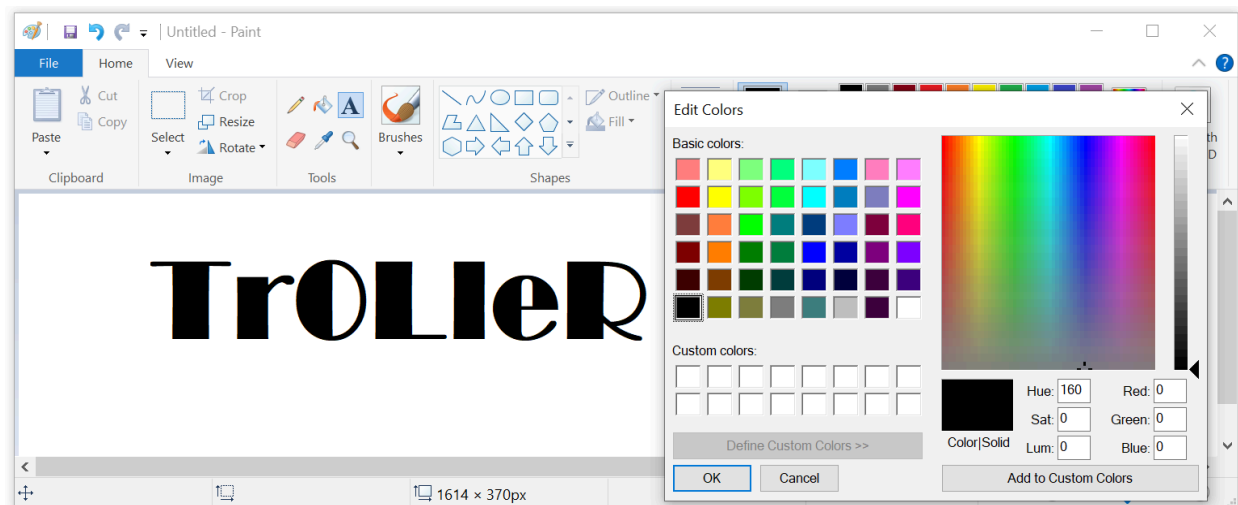
²³⁴ <https://ss64.com/nt/find.html>

mspaint.exe (Paint)

“mspaint.exe” is a PE binary located at “%windir%\System32\mspaint.exe”. On 64-bit systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\mspaint.exe”. Both of the versions are digitally signed by Microsoft. It is a simple graphic/drawing editor included as part of the Windows operating system since Windows 1.0. “mspaint.exe” different editing tools like brushes, shape generators, pens, eraser, color selection, bucket (fill with color) and magnifier²³⁵ - as shown in the screenshot below (It is the Windows 10 version).

Overall, “mspaint.exe” supports different image formats like: Windows bitmap (BMP), PNG, GIF, JPG and single-page TIFF. By the way, AI art generators (DALL-E based) are going to be part of Microsoft Paint²³⁶.

Moreover, support for layers (adding/removing/merging/duplicating/etc) and support for opening/saving transparent PNG files had been added to paint²³⁷. Those features fit together with the ability to remove the background of an image²³⁸. Lastly, we can check out the reference implementation of “mspaint.exe” as part of ReactOS²³⁹.



²³⁵ <https://mspaint.humanhead.com/#local:bd525d07a1f88>

²³⁶ https://en.wikipedia.org/wiki/Microsoft_Paint

²³⁷ <https://www.theverge.com/2023/9/18/23879221/microsoft-paint-testing-layers-transparency-photoshop-features>

²³⁸ <https://www.theverge.com/2023/9/7/23863377/microsoft-paint-background-removal-tool>

²³⁹ <https://github.com/reactos/reactos/tree/master/base/applications/mspaint>

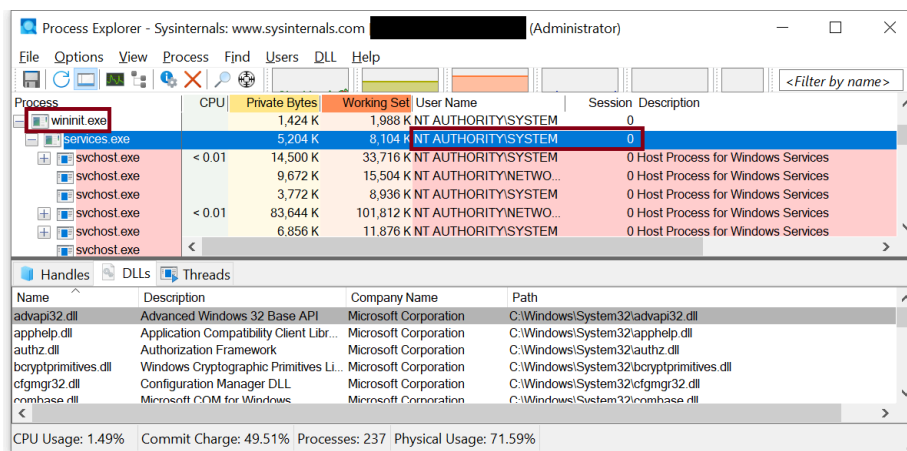
services.exe (Service Control Manager)

“services.exe” is a PE binary located at “%windir%\System32\services.exe”. It is part of the “Service Control Manager” (SCM), it provides an RPC (Remote Procedure Call) server (“RPC Control\ntsvcs”). By leveraging it, programs can manipulate and configure Windows services²⁴⁰ locally or remotely²⁴¹. A reference implementation of “services.exe” can be found as part of ReactOS²⁴².

Overall, “services.exe” is started when Windows starts. It is launched by “wininit.exe”²⁴³ on session 0 and is executed with the permissions and privileges of the “NT AUTHORITY\SYSTEM” (S-1-5-18) aka “Local System”. The binary is digitally signed by Microsoft. There are two built-in major tools for communicating with the SCM: “sc.exe” and the MMC snap-in “services.msc”²⁴⁴.

Moreover, the SCM provides an interface for performing various tasks as described next. Starting services/drivers on startup/demand. Maintaining/locking/unlocking the database of installed services (HKLM\SYSTEM\CurrentControlSet\Services). Transmitting control requests for running services. Maintaining the status of running drivers and services²⁴⁵.

Lastly, it should be executed only once on a Windows system regardless of the number of logged in users. By the way, on 64-bit systems unlike other Windows binaries (like “cmd.exe”) we don’t have a parallel 32-bit version of “services.exe”. We can also use the Win32 API for manipulating services²⁴⁶. The client-side API for the SCM is implemented as part of “%windir%\system32\advapi32.dll”²⁴⁷.



²⁴⁰ <https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4>

²⁴¹ https://publik.tuwien.ac.at/files/publik_273621.pdf

²⁴² <https://github.com/reactos/reactos/tree/master/base/system/services>

²⁴³ <https://medium.com/@boutnaru/the-windows-process-journey-wininit-exe-windows-start-up-application-5581bfe6a01e>

²⁴⁴ <https://medium.com/@boutnaru/the-windows-process-journey-mmc-exe-microsoft-management-console-a584afe66d86>

²⁴⁵ <https://learn.microsoft.com/en-us/windows/win32/services/service-control-manager>

²⁴⁶ <https://learn.microsoft.com/en-us/windows/win32/api/winsvc/>

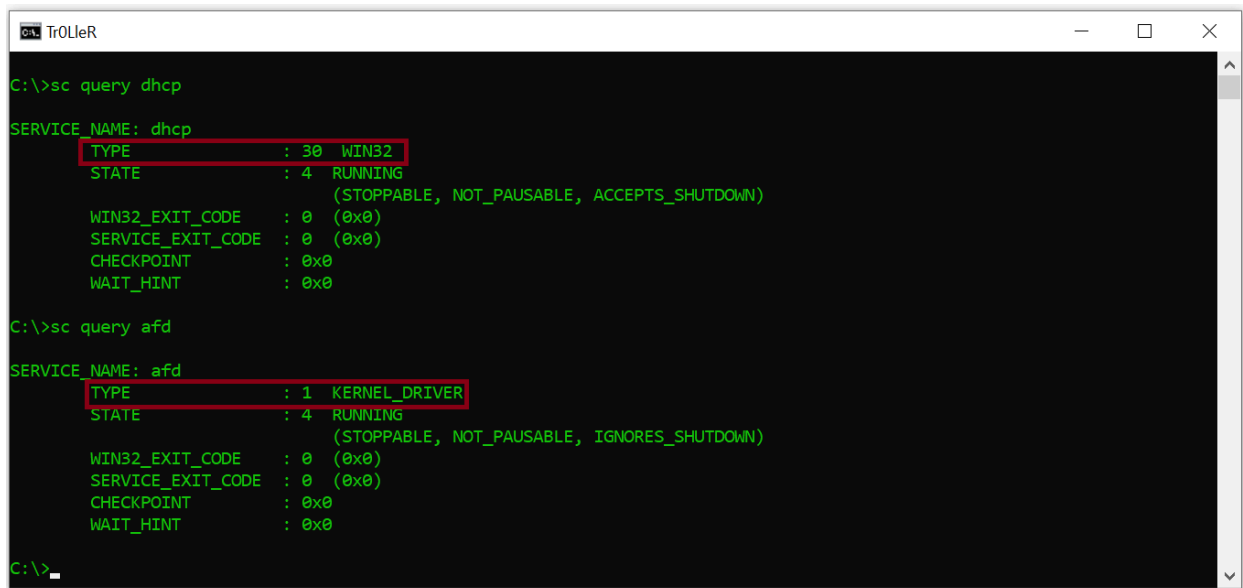
²⁴⁷ https://renenyffenegger.ch/notes/Windows/dirs/Windows/System32/services_exe/index

sc.exe (Service Control Manager Configuration Tool)

“sc.exe” is a PE binary located at “%windir%\System32\sc.exe”. By the way, on 64-bit systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\sc.exe”. Both files are digitally signed by Microsoft.

Overall, “sc.exe” is used to create/stop/start/query/delete/pause/configure/etc any Windows service²⁴⁸. For example, “sc.exe query <servicename>” is done by reading a subkey/entries of the service in the SCM (Service Control Manager) database - as shown in the screenshot below²⁴⁹. The SCM database is located in the registry in the following location: “HKLM\SYSTEM\CurrentControlSet\Services”.

Moreover, there are other command line options that can be used with “sc.exe” such as (but not limited to) viewing the security descriptor of the service (“sdshow”), showing/changing the description (“qdescription/description”), displaying/modifying the actions that are taken by the service in case of a failure (“qfailure/failure”), showing dependencies (“EnumDepend”) and creating/deleting a service (“create/delete”). By the way, “sc.exe” is also used for managing drivers, which are defined as services which execute in kernel mode - as shown in the screenshot below - more on that in future writeups²⁵⁰. Lastly, we can go over a reference implementation of “sc.exe” which is part of ReactOS²⁵¹.



```
C:\>sc query dhcp

SERVICE_NAME: dhcp
        TYPE               : 30  WIN32
        STATE                : 4  RUNNING
                        (STOPPABLE, NOT_PAUSABLE, ACCEPTS_SHUTDOWN)
        WIN32_EXIT_CODE      : 0  (0x0)
        SERVICE_EXIT_CODE   : 0  (0x0)
        CHECKPOINT          : 0x0
        WAIT_HINT           : 0x0

C:\>sc query afd

SERVICE_NAME: afd
        TYPE               : 1  KERNEL_DRIVER
        STATE                : 4  RUNNING
                        (STOPPABLE, NOT_PAUSABLE, IGNORES_SHUTDOWN)
        WIN32_EXIT_CODE      : 0  (0x0)
        SERVICE_EXIT_CODE   : 0  (0x0)
        CHECKPOINT          : 0x0
        WAIT_HINT           : 0x0

C:\>
```

²⁴⁸ <https://medium.com/@boutnaru/windows-services-part-2-7e2bdab5bce4>

²⁴⁹ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/sc-query>

²⁵⁰ <https://ss64.com/nt/sc.html>

²⁵¹ <https://github.com/reactos/reactos/tree/master/base/applications/sc>

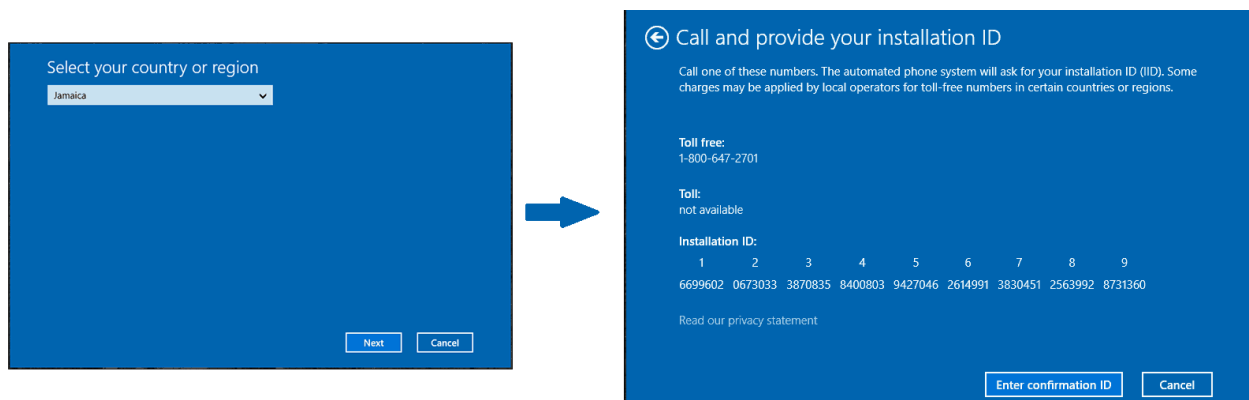
phoneactivate.exe (Phone Activation UI)

“phoneactivate.exe” is a PE binary located at “%windir%\System32\phoneactivate.exe”. Unlike other binaries there is no 32-bit version of it in Windows 64-bit systems (as we have with “cmd.exe” for example). The binary is digitally signed by Microsoft.

Overall, we can activate Windows using an internet connection (aka Online activation). Also, we can activate Windows by phone. In this case we try activating our device over the phone, this connects us to Microsoft support for our region and country²⁵².

Thus, the goal of “phoneactivate.exe” is to provide the phone activation UI (User Interface). One common use case for using it is if the Windows license was used in another computer. After the phone activation is launched we need to choose our country and select next - as shown in the screenshot below. Then, using the phone numbers shown on the screen we can call the support agent and provide the installation ID - also shown in the screenshot below²⁵³.

Lastly, after verifying the product key and using the installation ID the agent will provide a confirmation ID for activating Windows. By the way, we can also launch “Contact Support” and use a chat versus calling.



²⁵²<https://support.microsoft.com/en-us/windows/product-activation-for-windows-online-support-telephone-numbers-35f6a805-1259-88b4-f5e9-b52ccef91a0>

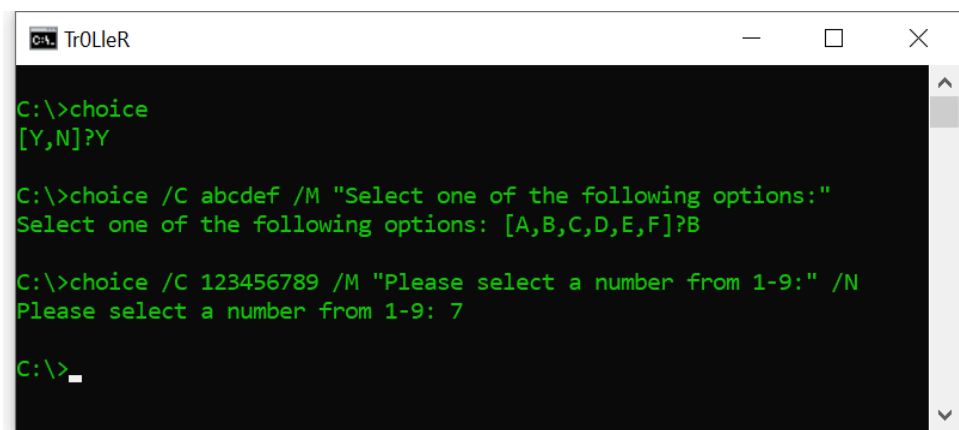
²⁵³<https://www.groovypost.com/howto/save-windows-10-spotlight-lock-screen-pictures/>

choice.exe (Offers the User a Choice)

“choice.exe” (Offers the user a choice) is a PE binary located at “%windir%\system32\choice.exe”. It is used for allowing users to select one (single key pressed) item from a list of choices, it returns the index of the selected choice. By default, we can choose between “Y” or “N” - as shown in the screenshot below²⁵⁴.

Moreover, we can customize the list of options and a text shown to the user using the different switches of “choice.exe” (“/C” and “/M” respectively) - as shown in the screenshot below. There are also other switches that allow us to control behavior of the command like: specify if the choices are case-sensitive (“/CS”), timeout for selecting one of the choices (“/T”) and more²⁵⁵.

Lastly, on 64-bit systems there is also a 32-bit version of “choice.exe” located at “%windir%\SysWOW64\choice.exe”. Both the 64-bit version and the 32-bit version are digitally signed by Microsoft.



```
C:\>choice
[Y,N]?Y

C:\>choice /C abcdef /M "Select one of the following options:"
Select one of the following options: [A,B,C,D,E,F]?B

C:\>choice /C 123456789 /M "Please select a number from 1-9:" /N
Please select a number from 1-9: 7

C:\>_
```

²⁵⁴ <https://ss64.com/nt/choice.html>

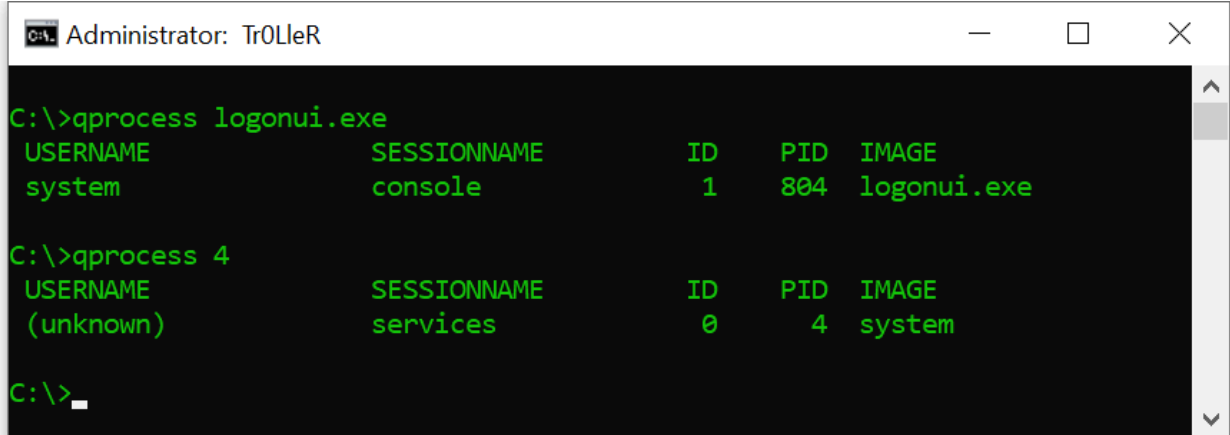
²⁵⁵ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/choice>

qprocess.exe (Query Process Utility)

“qprocess.exe” is a PE binary located at “%windir%\System32\qprocess.exe”. It is used for displaying information about processes. Also, it supports displaying information about processes that have been executed on a Remote Desktop Session Host Server ²⁵⁶.

Moreover, as opposed to other executables like “cmd.exe”²⁵⁷, on 64-bit versions of Windows there is no 32-bit version of “qprocess.exe”. The binary itself is digitally signed by Microsoft.

Lastly, “qprocess.exe” provides different command line switches. Using them we can list all processes for all sessions (“*”), display processes based on/process id/username/session name/session ID/program name²⁵⁸ - as shown in the screenshot below.



```
C:\>qprocess logonui.exe
USERNAME      SESSIONNAME   ID    PID  IMAGE
system        console       1     804  logonui.exe

C:\>qprocess 4
USERNAME      SESSIONNAME   ID    PID  IMAGE
(unknown)    services     0     4   system

C:\>_
```

²⁵⁶ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/qprocess>

²⁵⁷ <https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b>

²⁵⁸ <https://ss64.com/nt/query-process.html>

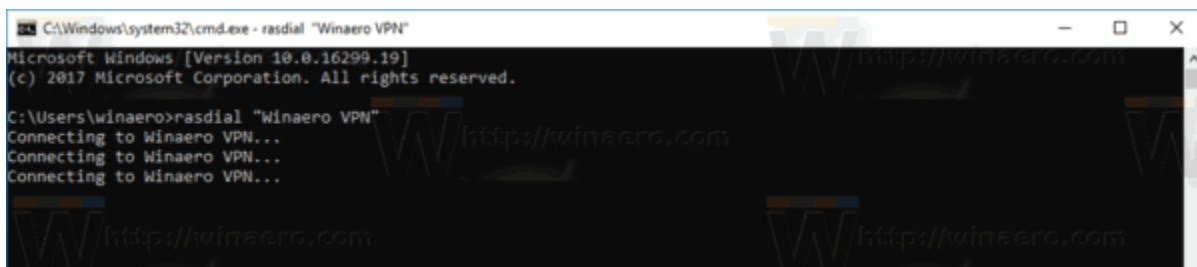
rasdial.exe (Remote Access Command Line Dial UI)

“rasdial.exe” is a PE binary located at “%windir%\System32\rasdial.exe”. It is used for connecting/disconnecting from a VPN (Virtual Private Network)/dial up connection²⁵⁹.

Overall, on 64-bit versions of Windows there is also a 32-bit version of the binary located at “%windir%\SysWOW64\rasdial.exe”. Both the 64-bit version and the 32-bit version are digitally signed by Microsoft.

Moreover, using the command line switches of “rasdial.exe” we can provide different information for a connection. Examples are : a username for connection, a password, a phone number to connect and a callback number. In case we execute “rasdail.exe” without any arguments the status of the current connection is displayed²⁶⁰.

Lastly, to specify credentials (username and password) we can execute the following command: “ rasdial ‘ConnectionName’ ‘Username’ ‘Password’ ”²⁶¹.



```
Microsoft Windows [Version 10.0.16299.19]
(c) 2017 Microsoft Corporation. All rights reserved.

C:\Users\winaero>rasdial "Winaero VPN"
Connecting to Winaero VPN...
Connecting to Winaero VPN...
Connecting to Winaero VPN...
```

²⁵⁹ [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/ff859533\(v=ws.11\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2012-r2-and-2012/ff859533(v=ws.11))

²⁶⁰ <https://ss64.com/nt/rasdial.html>

²⁶¹ <https://gist.github.com/stormwild/ec0898fe8bf25f58f4a6bf2576dc5e3f>

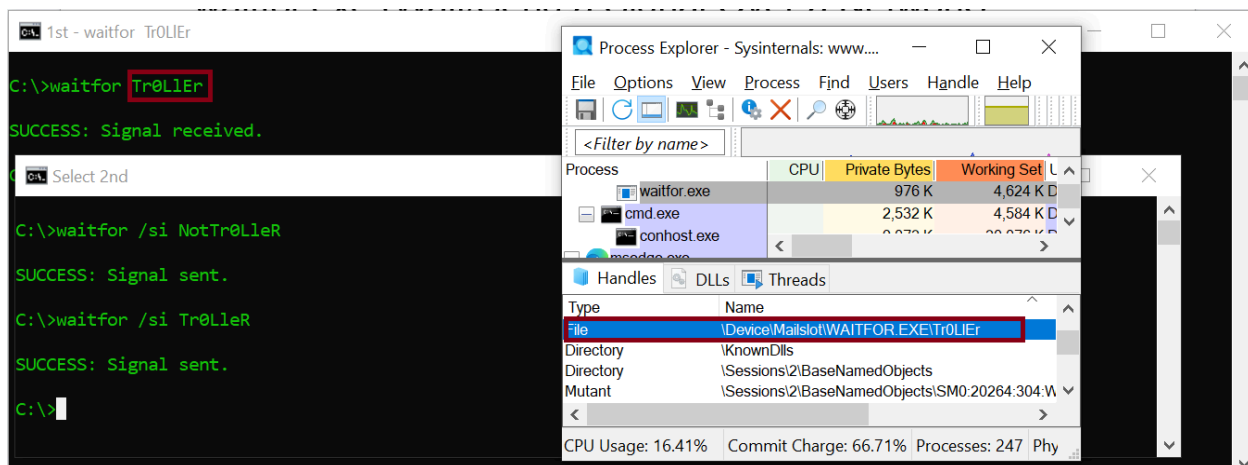
waitfor.exe (Wait/Send a Signal Over a Network)

“waitfor.exe” is a PE binary located at “%windir%\System32\waitfor.exe”. It is used for sending/waiting for a signal on a system. We can also use “waitfor.exe” in order to synchronize between computer systems over the network²⁶². By the way, on 64-bit systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\waitfor.exe”. Both the 32-bit version and the 64-bit version are digitally signed by Microsoft.

Overall, “waitfor.exe” is based on the mailslot²⁶³ IPC mechanism. When selecting a name for a signal to wait for, it is used as part of the naming of the mailslot using the following format “\\.\mailslot\WAITFOR.EXE\[SIGNAL NAME]” - as shown in the screenshot below. The signal itself is not case sensitive (the same as files in Windows).

Moreover, when using “waitfor.exe” for remote synchronization we can provide the username/password for authentication using the command line switches (“/u” and “/p” respectively) and “/” for providing the name/IP of the remote system²⁶⁴.

Lasly, we can think of “waitfor.exe” as a combination of the Linux commands “kill”²⁶⁵ and the “trap” command²⁶⁶. The first can send signals and the second one can wait for signals. Also, “tap” can be implemented in different ways such as a builtin command of a shell.



²⁶² <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/waitfor>

²⁶³ <https://medium.com/@boutnaru/the-windows-concept-jou-d35f84d8cc02>

²⁶⁴ <https://ss64.com/nt/waitfor.html>

²⁶⁵ <https://man7.org/linux/man-pages/man1/kill.1.html>

²⁶⁶ <https://man7.org/linux/man-pages/man1/trap.1p.html>

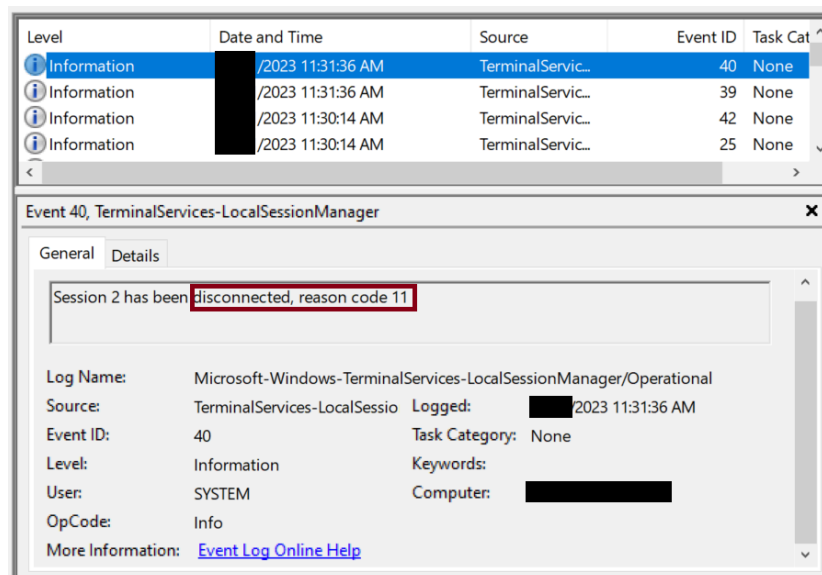
tsdiscon.exe (Session Disconnection Utility)

“tsdiscon.exe” is a PE binary located at “%windir%\System32\tsdiscon.exe”. It is used for disconnecting from a remote desktop services session. By the way, on 64-bit systems unlike other binaries like “cmd.exe”²⁶⁷ there is not 32-bit version of “tsdiscon.exe” in parallel to the 64-bit version.

Overall, using different switches we can specify the ID of the session or the session name that we want to disconnect. Also, we can provide the name of the terminal server containing the session we want to disconnect (“/server:<SERVER_NAME>”). By the way, if we don’t provide any session ID/name the current session is going to be disconnected²⁶⁸.

Moreover, there should not be any data loss when disconnecting from a session. The applications are still running, thus we can reconnect to the session. We must have full control permissions/disconnect permissions in order to disconnect another user from a session²⁶⁹. This can also be done for sessions within a virtual machine.

Lastly, when executing “tsdiscon.exe” an event is logged (ID 40) in the event viewer under the following location “Applications and Services Logs -> Microsoft -> Windows -> TerminalServices-LocalSessionManager -> Operational” - as shown in the screenshot below. By the way, “reason code 11” means the user disconnecting from the session initiates the disconnection²⁷⁰.



²⁶⁷ <https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b>

²⁶⁸ <https://ss64.com/nt/tsdiscon.html>

²⁶⁹ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/tsdiscon>

²⁷⁰ <https://www.anyviewer.com/how-to/session-has-been-disconnected-reason-code-0-2578.html>

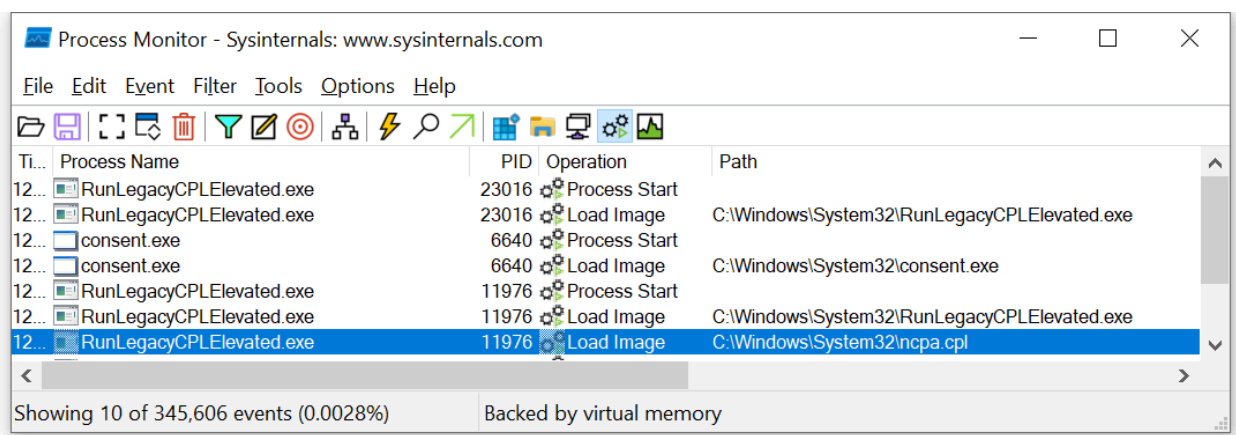
RunLegacyCPLElevated.exe (Running Legacy Control Panel Applet in Elevated Mode)

“RunLegacyCPLElevated.exe” is a PE binary located at “%windir%\System32\RunLegacyCPLElevated.exe”. It is used for running a legacy control panel applet in elevated mode. On 64-bit Windows systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\RunLegacyCPLElevated.exe”. By the way, both binaries are digitally signed by Microsoft.

Overall, we should execute “RunLegacyCPLElevated.exe” using the following arguments “RunLegacyCPLElevated.exe shell32.dll, Control_RunDLL <CPL_FILE_PATH_TO_LOAD>”. An example of execution is “RunLegacyCPLElevated.exe shell32.dll, Control_RunDLL %windir%\system32\ncpa.cpl” - as shown in the screenshot below.

Moreover, when executing the binary the chain of execution is as follows: “RunLegacyCPLElevated.exe” performs an RPC call to execute “consent.exe”²⁷¹, which is started by the “Application Information” service (hosted by svchost.exe). After that “RunLegacyCPLElevated.exe” is executed again with the same arguments using the elevated access token, this is the process that loads and executes the function for the “*.cpl” file - as shown in the screenshot below.

Lastly, we can think about “RunLegacyCPLElevated.exe” as a “rundll32.exe”²⁷² which starts the control panel applet with high permissions. Thus, it is similar to executing (without the elevation part) to “rundll32.exe shell32.dll, Control_RunDLL %windir%\system32\ncpa.cpl”.



²⁷¹ <https://medium.com/@boutnaru/the-windows-process-journey-consent-exe-consent-ui-for-administrative-applications-d8e6976e8e40>

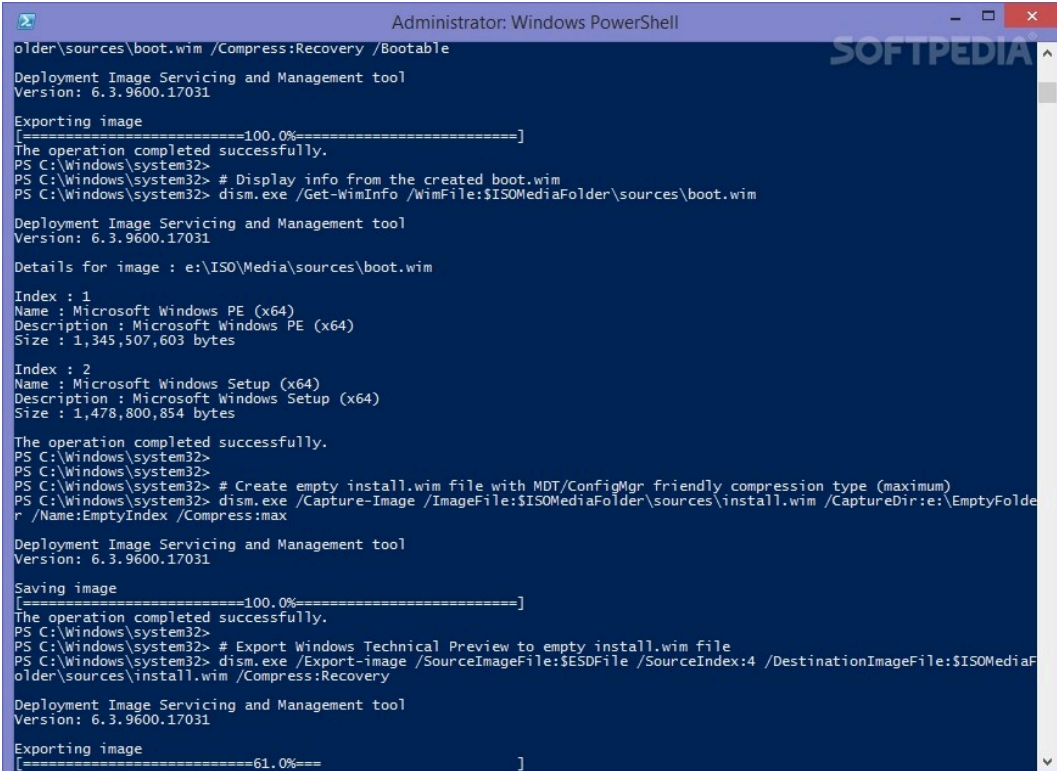
²⁷² <https://medium.com/@boutnaru/the-windows-process-journey-rundll32-exe-windows-host-process-415132f1363>

dism.exe (Deployment Image Servicing and Management Tool)

“dism.exe” is a PE binary located at “%windir%\System32\dism.exe”. We can use it in order to enumerate/install/uninstall/configure/update features and packages as part of the Windows operating system²⁷³. On 64 bit systems there is also a 32-bit version of the binary located at “%windir%\SysWOW64\Dism.exe”. Both binaries are digitally signed by Microsoft.

Overall, “dism.exe” can be used to prepare/service “Windows Images” that can be used for Windows PE/Windows RE (Recovery Environment)/Windows Setup. It can also service “*.wim” (Windows Image) files or “*.vhd”/“*.vhdx” (virtual hard disks) files²⁷⁴.

Lastly, “dism.exe” can be executed with elevated permissions which allows parsing of information of image files and saving changes - as shown in the screenshot below²⁷⁵. Thus, “dism.exe” can modify offline image files in the different ways such as: ways: add language packs, add package updates, enable/disable OS features, combine images, adding device drivers²⁷⁶.



```
Administrator: Windows PowerShell
older\sources\boot.wim /Compress:Recovery /Bootable
Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Exporting image
[=====100.0%=====]
The operation completed successfully.
PS C:\Windows\system32>
PS C:\Windows\system32> # Display info from the created boot.wim
PS C:\Windows\system32> dism.exe /Get-WimInfo /WimFile:$ISOMediaFolder\sources\boot.wim

Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Details for image : e:\ISO\Media\sources\boot.wim

Index : 1
Name : Microsoft Windows PE (x64)
Description : Microsoft Windows PE (x64)
Size : 1,345,507,603 bytes

Index : 2
Name : Microsoft Windows Setup (x64)
Description : Microsoft Windows Setup (x64)
Size : 1,478,800,854 bytes

The operation completed successfully.
PS C:\Windows\system32>
PS C:\Windows\system32>
PS C:\Windows\system32> # Create empty install.wim file with MDT/ConfigMgr friendly compression type (maximum)
PS C:\Windows\system32> dism.exe /Capture-Image /ImageFile:$ISOMediaFolder\sources\install.wim /CaptureDir:e:\EmptyFolder /Name:EmptyIndex /Compress:max

Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Saving image
[=====100.0%=====]
The operation completed successfully.
PS C:\Windows\system32>
PS C:\Windows\system32> # Export Windows Technical Preview to empty install.wim file
PS C:\Windows\system32> dism.exe /Export-image /SourceImageFile:$ESDFile /SourceIndex:4 /DestinationImageFile:$ISOMediaFolder\sources\install.wim /Compress:Recovery

Deployment Image Servicing and Management tool
Version: 6.3.9600.17031

Exporting image
[=====61.0%=====]
```

²⁷³ <https://ss64.com/nt/dism.html>

²⁷⁴ <https://learn.microsoft.com/en-us/windows-hardware/manufacture/desktop/what-is-dism?view=windows-11>

²⁷⁵ <https://shopperlasopa179.weebly.com/dismexe-wim.html>

²⁷⁶ <https://www.slideserve.com/akamu/cn1176-computer-support-powerpoint-ppt-presentation>

chkdsk.exe (Check Disk Utility)

“chkdsk.exe” (Check Disk Utility) is a PE binary located at “%windir%\System32\chkdsk.exe”. On 64-bit systems there is also a 32-bit version located at “%windir%\SysWOW64\chkdsk.exe”. It is used to check the file-system/file-system metadata of a volume for logical/physical errors. In order to execute it the user needs to be a member of the local administrator group²⁷⁷.

Moreover, “chkdsk.exe” can not only scan for errors but also fix some of them based on the different switches given when executing it. If no parameter was given it will run in read-only mode - as shown in the screenshot below. For fixing structural issues we can use “/f” and to try recovering data from corrupted parts of the physical drive we can also add “/r”. To dismount the drive for scanning and fixing we should use “/x”²⁷⁸.

Lastly, “chkdsk.exe” is a CLI tool which is digitally signed by Microsoft. When running a check “chkdsk.exe” performs 3 main stages: examination of basic filesystem structure, examination of file name linkage and examination of security descriptors - as shown in the screenshot below.



```
Administrator: Command Prompt
WARNING! /F parameter not specified.
Running CHKDSK in read-only mode.

Stage 1: Examining basic file system structure ...
 664064 file records processed.
File verification completed.
Phase duration (File record verification): 12.01 seconds.
 19116 large file records processed.
Phase duration (Orphan file record recovery): 0.00 milliseconds.
 0 bad file records processed.
Phase duration (Bad file record checking): 0.72 milliseconds.

Stage 2: Examining file name linkage ...
 172 reparse records processed.
 1053636 index entries processed.
Index verification completed.
Phase duration (Index verification): 24.83 seconds.
 0 unindexed files scanned.
Phase duration (Orphan reconnection): 5.88 seconds.
 0 unindexed files recovered to lost and found.
Phase duration (Orphan recovery to lost and found): 0.75 milliseconds.
 172 reparse records processed.
Phase duration (Reparse point and Object ID verification): 4.38 milliseconds.

Stage 3: Examining security descriptors ...
Security descriptor verification completed.
Phase duration (Security descriptor verification): 27.02 milliseconds.
 194787 data files processed.
Phase duration (Data attribute verification): 1.28 milliseconds.
CHKDSK is verifying Usn Journal...
 39281896 USN bytes processed.
Usn Journal verification completed.
Phase duration (USN journal verification): 465.19 milliseconds.

Windows has scanned the file system and found no problems.
No further action is required.

132529210 KB total disk space.
55475728 KB in 447427 files.
 297176 KB in 194788 indexes.
 0 KB in bad sectors.
 778278 KB in use by the system.
 65536 KB occupied by the log file.
75978028 KB available on disk.

 4096 bytes in each allocation unit.
33132302 total allocation units on disk.
18994507 allocation units available on disk.
Total duration: 43.24 seconds (4324 ms).
```

²⁷⁷ <https://learn.microsoft.com/en-us/windows-server/administration/windows-commands/chkdsk?tabs=event-viewer>

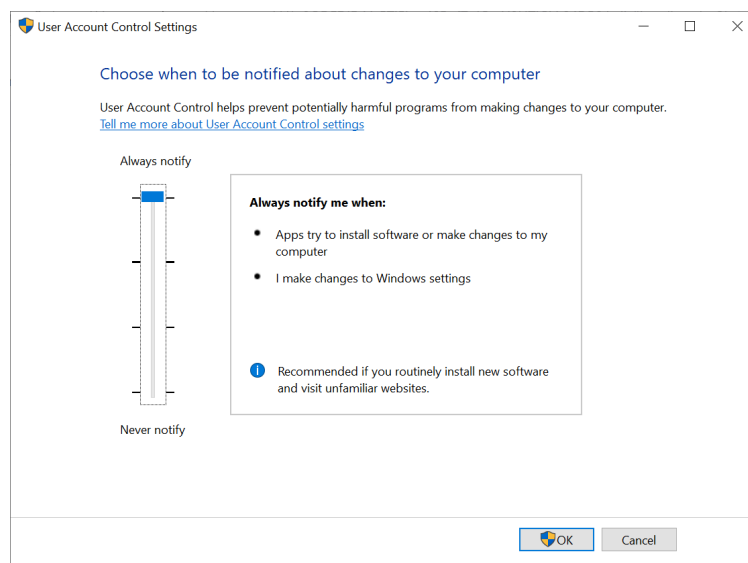
²⁷⁸ <https://www.avg.com/en/signal/how-to-use-chkdsk-windows>

UserAccountControlSettings.exe (Configuring UAC Settings)

“UserAccountControlSettings.exe” is a PE binary file located at “%windir%\system32\UserAccountControlSettings.exe”. On 64-bit systems there is also a 32-bit version of the file located at “%windir%\SysWOW64\UserAccountControlSettings.exe”. It is used in order to change the settings of UAC (User Account Control)²⁷⁹. The binary is digitally signed by Microsoft.

Overall, “UserAccountControlSettings.exe” allows a user to select the level of notifications in case apps try to install software/change computer settings or whether the user itself tries to do those things²⁸⁰. There are a total of four levels that we can select from (using the slider) - as shown in the screenshot below.

First, the lower one is to never notify (whether app/user is trying to install software making changes to Windows settings). Second, notify only if apps are trying to make changes (not relevant if the user does that), by the way the desktop won't be dimmed. Third, as the previous but dims the desktop (meaning using the secure desktop), it is also the default setting. Fourth, notify if an app/user is trying to install software/make changes to the Windows settings.



²⁷⁹ https://renenyffenegger.ch/notes/Windows/dirs/Windows/System32/UserAccountControlSettings_exe

²⁸⁰ <https://www.elevenforum.com/t/change-user-account-control-uac-settings-in-windows-11.1523>

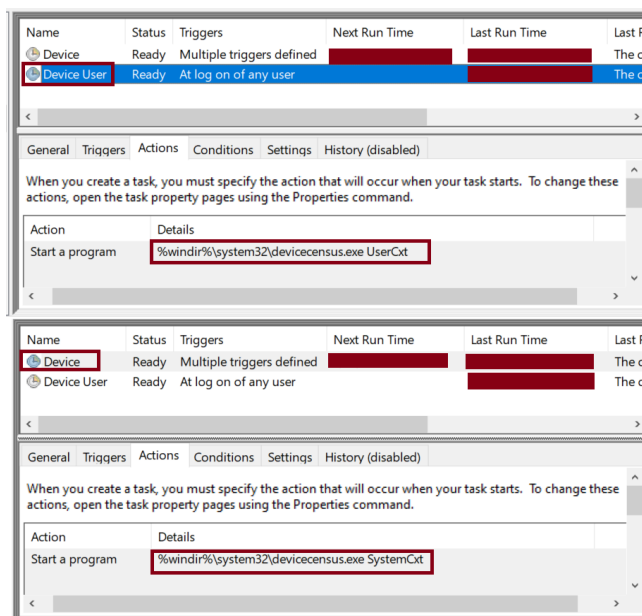
DeviceCensus.exe (Device Information)

“DeviceCensus.exe” is a PE binary located at “%windir%\System32\DeviceCensus.exe”. As opposed to other executables such as “cmd.exe”²⁸¹ there is only a 64-bit version of “DeviceCensus.exe” as part of a 64-bit version of Windows (no parallel 32-bit version). By the way, the binary is digitally signed by Microsoft.

Overall, “DeviceCensus.exe” is executed by the “Task Scheduler”²⁸² on Windows. There are two tasks which are configured by default to run “DeviceCensus.exe”: “Device” and “Device User”. Both of them can be found in the following location in the “Task Scheduler Library”: “Microsoft\Windows\Device Information” - as shown in the screenshot below. The second one is executed at log on of every user.

Moreover, “DeviceCensus.exe” accepts as command line arguments the following: “SystemCxt” (used by the “Device” task) and “UserCxt” (used by the “Device User” task). Each flow which is triggered based on them calls exported functions from “%windir%\system32\dcntel.dll”. The first one calls the “RunSystemContextCensus” function and the second calls the “RunUserContextCensus” function.

Lastly, based on different documentation “DeviceCensus.exe” helps Microsoft improve user experience by understanding how their products are being used. It is used to collect information like hardware in use, performance data and most used features. Thus, it is part of telemetry data collection in Windows²⁸³.



²⁸¹ <https://medium.com/@boutnaru/the-windows-process-journey-cmd-exe-windows-command-processor-501be17ba81b>

²⁸² <https://medium.com/@boutnaru/windows-scheduler-tasks-84d14fe733c0>





²⁸³ <https://www.file.net/process/devicecensus.exe.html>

MpCmdRun.exe (Microsoft Malware Protection Command Line Utility)

“MpCmdRun.exe” is a PE binary located at “C:\ProgramData\Microsoft\Windows Defender\Platform\[VERSION]\MpCmdRun.exe”. By the way, [VERSION] matches the file version stored in the PE. Its description states it is the “Microsoft Malware Protection Command Line Utility”. Also, the binary is also digitally signed by Microsoft. By the way, it is also called “Microsoft Defender Antivirus command-line utility” as part of the Microsoft documentation²⁸⁴. It is used as a command line frontend for “Microsoft Malware Protection”.

Moreover, by default there are four Windows schedule tasks²⁸⁵ which are based on “MpCmdRun.exe” as their action: “Windows Defender Cache Maintenance” (periodic maintenance task), “Windows Defender Cleanup” (periodic cleanup task), “Windows Defender Scheduled Scan” (periodic scan task) and “Windows Defender Verification” (periodic verification task) - as shown in the screenshot below. We can find all of them in the following location : “Task Scheduler Library->Microsoft->Windows->Windows Defender”.

Lastly, “MpCmdRun.exe” has multiple command line arguments supported in different categories such as scanning and tracing. We can get information about all the available options using the “-h” switch or the “?”.

Name	Status
 Windows Defender Cache Maintenance	Ready
 Windows Defender Cleanup	Ready
 Windows Defender Scheduled Scan	Ready
 Windows Defender Verification	Ready

²⁸⁴<https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus-windows?view=o365-worldwide>

²⁸⁵ <https://medium.com/@boutnaru/windows-scheduler-tasks-84d14fe733c0>

MpDefenderCoreService.exe (Antimalware Core Service)

“MpDefenderCoreService.exe” is a PE binary located at “C:\ProgramData\Microsoft\Windows Defender\Platform\[VERSION]\MpDefenderCoreService.exe”. By the way, [VERSION] matches the file version stored in the PE. Its description states it is the “Antimalware Core Service”. Also, the binary is also digitally signed by Microsoft.

Moreover, “MpDefenderCoreService.exe” can be used as the start image of “Microsoft Defender Antivirus Core service” (MdCoreSvc). Its goal is to improve the stability and performance of “Windows Defender Antivirus”²⁸⁶. The separation to different services is was not since the creation of “Microsoft Defender Antivirus” - as shown in the screenshot below²⁸⁷.

Lastly, we can think about it as part of the processes of “Microsoft Defender Antivirus”²⁸⁸ together with processes like: “NisSrv.exe”²⁸⁹ and “MsMpEng.exe”.

MC687846 — (Updated) New Microsoft Defender Antivirus services on Windows Devices



>60 Days

Updated November 30, 2023: We have updated the rollout timeline below. Thank you for your patience.

Microsoft Defender Antivirus on Windows 10 and Windows 11 will be shipping with two new services:

- Microsoft Defender Core service.
- Microsoft Data Loss Prevention Service

²⁸⁶<https://github.com/MicrosoftDocs/microsoft-365-docs/blob/public/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus-windows.md>

²⁸⁷<https://techcommunity.microsoft.com/t5/public-sector-blog/december-2023-microsoft-365-us-public-sector-roadmap-newsletter/ba-p/4010161>

²⁸⁸<https://learn.microsoft.com/en-us/microsoft-365/security/defender-endpoint/microsoft-defender-antivirus-windows?view=o365-worldwide>

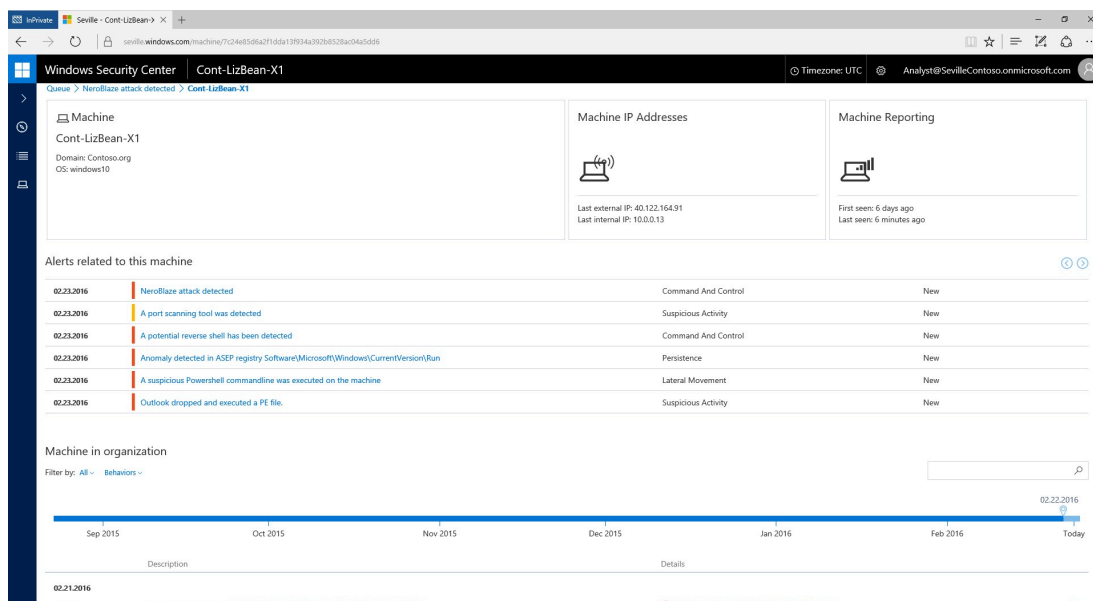
²⁸⁹<https://medium.com/@boutnaru/the-windows-process-journey-nissrv-exe-microsoft-network-realtime-inspection-service-48b1245f434c>

MsSense.exe (Windows Defender Advanced Threat Protection Service Executable)

“MsSense.exe” (Windows Defender Advanced Threat Protection Service Executable) is a PE binary located at “%ProgramFiles%\Windows Defender Advanced Threat Protection\MsSense.exe”. It is used as the main binary of the “Windows Defender Advanced Threat Protection Service” (Sense). The description of the services states “Windows Defender Advanced Threat Protection service helps protect against advanced threats by monitoring and reporting security events that happen on the computer”.

Moreover, the service is executed using the permissions/privileges of the “Local System” user²⁹⁰. By the way, “MsSense.exe” is digitally signed by Microsoft. It is dependent on “MsSense.dll” (Windows Defender Advanced Threat ProtectionSense Library), which by default is located in the same directory as “MsSense.exe”.

Lastly, the goal of “Windows Defender Advanced Threat Protection” is to help detect, investigate and respond to advanced attacks (focused on enterprises). This is done by providing key information about who/what/why the attack happened - as shown in the screenshot below. Also, it provides response recommendations and time-travel like capabilities (6-months historical data on state of the machine) - as shown in the screenshot below²⁹¹.



²⁹⁰ <https://medium.com/@boutnaru/the-windows-security-journey-local-system-nt-authority-system-f087dc530588>

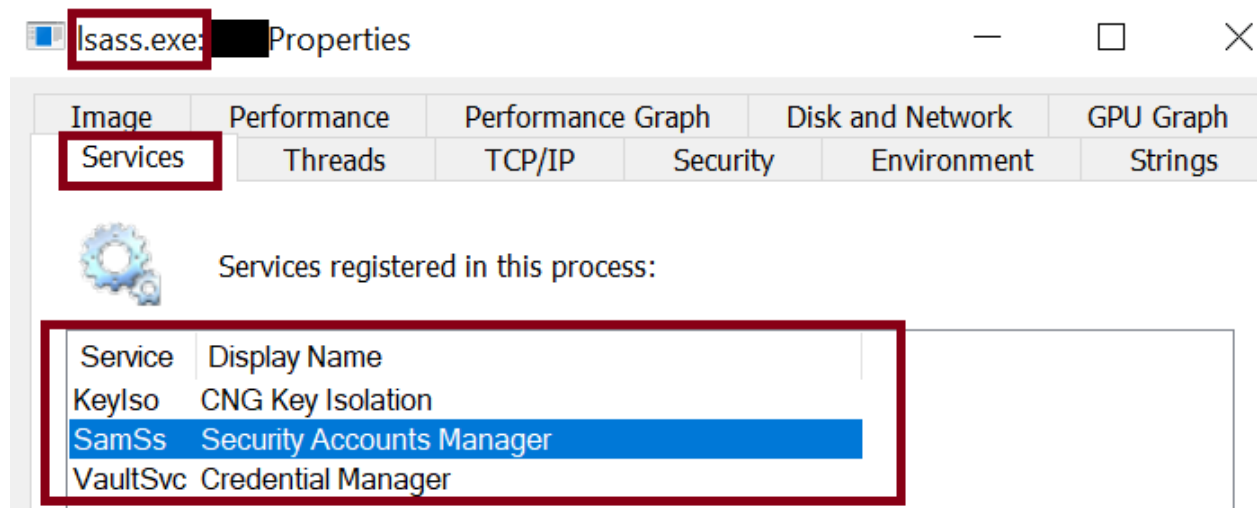
²⁹¹ <https://blogs.windows.com/windowsexperience/2016/03/01/announcing-windows-defender-advanced-threat-protection/>

lsass.exe (Local Security Authority Process)

“lsass.exe” (Local Security Authority Subsystem Service) is a PE binary located in “%windir%\System32\lsass.exe”. It is used for enforcing security policy, creating access tokens for logging on users, writing the security event log and more²⁹².

Moreover, “lsass.exe” can hold valuable authentication data like: kerberos tickets (TGT/TGS), LM/NT hashes, encrypted password and more²⁹³. Thus, Because “lsass.exe” stores the current user OS credentials (and can even store domain admin credentials in some cases). Due to that, it is an appealing target for attacks which can allow them to perform lateral movement. For hardening “lsass.exe” administrators can: enable it as PPL, enable credential guard, enable restricted admin mode for RDP and disable WDigest logon²⁹⁴.

Lastly, the “lsass.exe” process is hosting different services inside its own process memory address space. We have “KeyIso” (CNG Key Isolation) which provides key process isolation to private keys and associated cryptographic operations as required by Common Criteria. ”SamSs” (Security Account Manager), the startup of this service signals other services that the SAM is ready to accept requests. “VaultSvc” (Credential Manager), which is used to provide secure storage and retrieval of credentials to users/applications/security service packages - as shown in the screenshot below (taken from Process Explorer). By the way, if the computer is joined into a domain there will also be a service for network logon.



²⁹² https://en.wikipedia.org/wiki/Local_Security_Authority_Subsystem_Service

²⁹³ <https://redcanary.com/threat-detection-report/techniques/lsass-memory/>

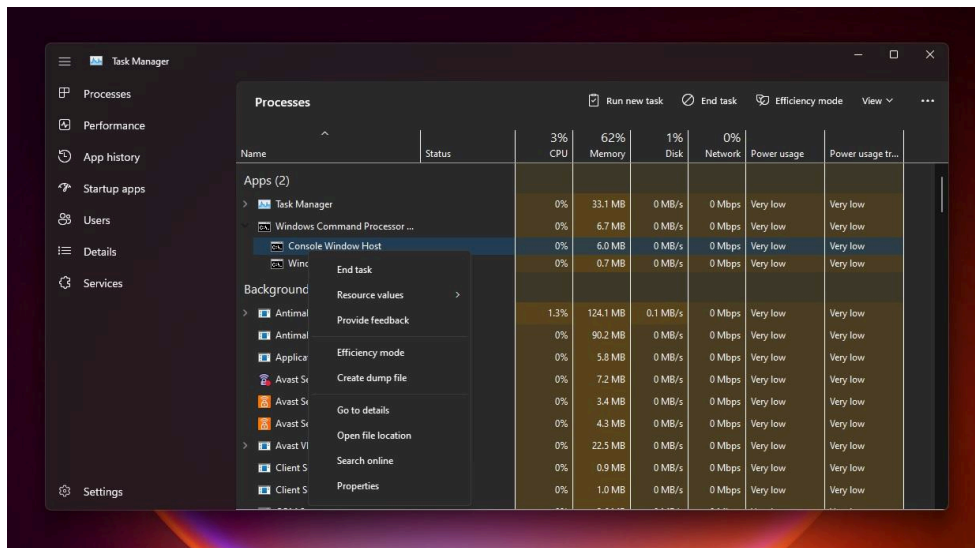
²⁹⁴ <https://www.microsoft.com/en-us/security/blog/2022/10/05/detecting-and-preventing-lsass-credential-dumping-attacks/>

Taskmgr.exe (Task Manager)

“Taskmgr.exe” (Task Manager) is a PE binary located in “%windir%\system32\Taskmgr.exe”. It can be used in order to view/manage current running processes, view system resources usage, analyze performance and close unresponsive applications by leveraging its user interface²⁹⁵. The binary is digitally signed by Microsoft.

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 devices²⁹⁶.

Lastly, we can go over a reference implementation of “taskmgr.exe” as part of ReactOS²⁹⁷. 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”²⁹⁸. By the way, based on the command line arguments passed to “taskmgr.exe” we can identify the way in which it was launched²⁹⁹.



²⁹⁵ <https://www.spyshelter.com/exe/microsoft-windows-taskmgr-exe/>

²⁹⁶ <https://www.bleepingcomputer.com/news/microsoft/hands-on-with-windows-11s-new-task-manager/>

²⁹⁷ <https://github.com/reactos/reactos/tree/master/base/applications/taskmgr>

²⁹⁸ <https://www.howtogeek.com/66622/stupid-geek-tricks-6-ways-to-open-windows-task-manager/>

²⁹⁹ <https://www.hexacorn.com/blog/2018/07/22/taskmgr-exe-slashing-numbers/>