

# MicroWinpeBuilder

## **Purpose:**

Learn to adapt yourself a Winpe10 64bits by adding, for example, MMC (eventlog, devices, disks, services, firewall, etc.), the desktop with the taskbar, Ncpa, Netsh, Wifi, MSI, file server, Audio, Wow64, adm session, Bits, Winrm, MSTCS with NLA, printer/scan USB or network, Bluetooth, Wpd/Mtp...

In this document, I gather all the useful information that I discovered myself or that I found on the reference sites:

<http://theoven.org>

<http://reboot.pro>

<https://msfn.org>

**The site of Slore** with whom I share a lot :

<https://gitee.com/slorelee/wimbuilder2>

Without the advice and valuable help of the participants of these sites, I would never have managed to build and adapt "my" Winpe according to my desire.

This document will hopefully be useful to curious beginners like me. I do not pretend to believe that an expert will find any interest in it.

**Limit:** I started this document with version 1511 build 10586. Windows versions follow each other quickly. The changes introduced by MS disrupt the precarious balance of a highly customized Winpe. So what's true for one version may be wrong for the next one. In addition, I can simply make mistakes or make bad statements.

Please contact me at the address below. I am interested in your comments, whatever they may be.

About me: I left school very early, as early as 18 years old. I never studied computer science at school. I started with «debug.exe» on DOS2.0. I don't know any subject matter. I know my knowledge is imperfect. I use IDA and WinDbg as a beginner. I am an eternal beginner and I know how to remain humble.

## A brief presentation of the goal

- Understanding how to adapt Winpe10 generated by ADK
  - It requires a starting point and basic knowledge, often provided by third parties. Various people over the years have touched on this topic. CF "my references."
  - You also have to make choices. Changing a finished product like ADK's Winpe implies that one has a need, a reason, a goal. Some will choose to reduce the size of the «Boot.Wim» file. Others, like me, will make the opposite choice, add as many features as possible.
  - To understand is also to make sometimes unsuccessful trials. In this context, efficiency and speed are not the main criteria.
- Learn to carry out basic investigations
  - The first adaptation is often to add "The Explorer Office." By searching through various sources, we find all the elements to collect and inject into the «Boot.Wim» file. But when trying to add a feature that is not available in reference sites, you are often forced to implement an investigation that requires a de-escalation, a disassembler, tools like Procmon... In addition, it may be useful to compare hives with other sources (as with Winpe FullFlat).
- Incorporate information discovered or made available on the internet
  - When you have the data you need to customize your Winpe, you have to inject it into the «Boot.Wim» file. The length of the file and key lists, whether or not you want to give certain items, all this makes it essential either to use a third-party product (which I refer to here as "Winpe Builder"), or to create its own "Winpe Builder". For my part, wishing to remain autonomous, I wrote mine in Powershell.
- Provide more information for the beginner
  - Collecting basic data is tedious. Often, the information you search for is hidden in the bottom of a "Winpe Builder" script or in the back of a forum.
  - So my idea is to share my information collection.

## Warning

My scripts offered here (or on one of the reference sites) allow you to build a WinPe. But because of my choices (from DotNet in particular), the size of the «Boot.Wim»file is greater than 900MB. But it doesn't matter how big this construction is for educational purposes.

## My choices

Since it is a matter of learning and understanding, we must not hide the means used.

- Rule 1: Do not use external software (so without third-party "Winpe Builder")
  - I only use PowerShell scripts that include C# if needed
- Rule 2: Use Install.Wim's Software hive (for simplicity's sake)
  - Over the course of the versions, I got tired of looking for the smallest element to add. I agree not to know everything and I am satisfied with the result visible on the screen.
- Rule 3: do not offer options, all additions are necessary (too many interdependencies)
- Rule 4: Include item lists in the main script (to limit the number of files)
- Rule 5: Allow new features to be added without trying to reduce the size of the «Boot.Wim»

file.

It is necessary to define the use that one will make of his "Winpe". For my part, I want to add as many features as possible. This implies that the «Boot.Wim» file will be very large. This vision of my "Winpe" goes the opposite of looking for a minimum size. This makes it possible to make structuring choices by favouring the use of the entire hive (software, drivers...). The flip-being is that we will never know the detail.

- Rule 6: Change the binary code of Winpe programs to simplify treatments when starting Winpe. With 20H2, I modify several files to bypass the tests of the "MiniNT" and "SysprepInProgress" keys, and also to activate certain features.

## ***A "builder" written in Powershell***

I really appreciate Powershell. The ability to include C-lines provides effective assistance when needed. But some limitations still forced me to use the following two software: Reg.exe and Robocopy.exe.

Reg :

- it makes it easier to copy key tree
- Key tree copy is fast

Robocopy :

- it allows you to copy directories whose names exceed the 256-character limit
- but it doesn't copy older files from the source. So the newer files in the destination are not modified, overwritten.

For ACL :

- I use commands Powershell and a few C#

## ***My scripts***

I deposit them on one of the reference sites so as not to overload this document.

They are so poorly written that I give here an entry point into my main script for a quick search:

Open my script "Traitement.ps1." Find "function step3." Read the sequence of actions to be chained.

Lists of items to include are included in the script.

If you open it with "powershell\_ise," tap into the control window:

```
$psIse.CurrentFile.Editor.ToggleOutliningExpansion
```

This will close the "region" blocks and allow you to better locate the part you are interested in.

## ***Too many details kill the information***

This document does not include all the details (keys, files, acl) present in the PS scripts.

Classifying and prioritizing information is a complex activity.

And I don't always have very clear ideas or very orderly ideas.

Read the scripts if you want to check for a file, a key...

## Document structure

The first part brings together all the useful information about WinPe.

- Useful keys
- The known pitfalls
- Details for some added features

The second part gives some ideas to get into the investigation.

- How to add a driver
- What investigative tools to use
- "Historical" information that is not useful (this document is also my notepad)

The third part provides an illustration implemented by the scripts.

Appendices or archives... Maybe... and waiting to throw them in the trash.

## Preamble

### *What is supposedly known here*

- read WinPe documentation available on MS websites
- Be familiar with the tools available in l'ADK, DISM, BCDEDIT, BCDBOOT, etc
- know how to change a BCD
  - use the command « /store »
  - create an input with the command « copy »
  - enable the boot menu to be displayed with « /set {bootmgr} displaybootmenu yes »
  - don't forget that the BCD for WinPe ISO contains keys for the RamDisk that are not present in the hard drive BCD after a normal installation of a normal Windows 10
- know how to build BCDs for VHDs.
  - VHDs are very convenient for investigations because the deposited files persist after restarting. Changes in hives do not persist.
- have read the scripts CopyPe.cmd and MakeWinpeMedia.cmd from ADK

### *The basis of a tutorial for adding the native office*

I made a structuring choice to build my Winpe (C.F. "My Choices"):

**I use the hive Software from «Install.Wim»**

With this choice, the main steps of construction are:

- Preparing the context: mounting Boot.Wim and Install.Wim
- Load the 6 source and destination hives ( software, system, default )
- Changing many ACL (files, registry)
- Delete «Runas» of APPID ( change the keys you need «APPID» )
- Replace C: by X: in the hives used
- Copy the desired register parts: but what list used?
- Inject ProductPolicy (it contains permissions/licenses for features)
- Add customization keys from your searches: which list is used?
- Unmount the hives

- Copy files from your searches: which list is used?
- Load with DISM the drivers you want

Giving an exhaustive list of files, registry keys, commands to launch, manipulations with the various tools used quickly becomes a gigantic job.

You will find all these steps and lists in my script "Traitement.ps1". It's all there! Nothing is hidden there.

**A tool seems essential to me if you want to create and customize your own WinPe.**

There are several. It's up to everyone to choose or create their own.

### ***Pitfalls and anomalies***

The versions of Winpe follow one another. New MS changes often disrupt "old" bypasses made to activate features.

### **Known pitfalls: list to be completed**

I met these during my first tests:

- Change the Acl of the key "OLE" for DragAndDrop
- Delete the "WallpaperHost.exe" file if it is present
- Delete the file "windows.immersiveshell.serviceprovider.dll" if it is present (fullflat)
- Add the "imageres.dll" file from win10 because Winpe's is smaller (depends on the version)
- Add the "Environment" key to Winpe's Default hive (if not ADM office disruption: to check)

### **Corrected anomalies in my Winpe 1511 (build10586): for the record!**

- Wpeinit hangs 5 minutes at Winpe start

The dll "policymanager.dll" was present but the key "software\micros... \policymanager" was missing.

- The start-up of the "coremessagingregistrar" service failed

Missing the key «software\micros...\securitymanager»

- The office icons were displayed one minute after the office

keys were missing in the «HKCU\..\explorer»

- The sound didn't work and the notification icon at the bottom right displayed "no loudspeakers."

An ACL forbade access to the key «...\MmDevices\Audio\Render\...\properties» for accounts used by the service AudioSrv.

- Impossible move of office icons

This came from the "software\microsoft\ole" key that did not contain the information related to DragAndDrop. The loading of the new values failed because of the Acl of the key.

- Impossible shortcut creation on desktop

Missing the files «appwiz.cpl» and «osbaseln.dll» and their «.mui»

- Lack of wallpaper: it lacked productOptions-productPolicy and various keys
- snippingtools was not working: it was missing productOptions\productPolicy
- The right click on the "display, personalization" wallpaper doesn't launch anything: requires the 32-bit subsystem 🍷 **NO OK from V 1709**
- Mstsc: OK with NLA,
- Bits : a lot of changes to make it operational
- When logging in "adm," an error message from "Logonui.exe" appears but is not blocking: Winpe's "Imageres.dll" file is smaller than Windows. Check this point for new releases.

### Anomaly appeared with version 1607 build 14393

- WMP only reads MP3 format: corrected with script V17
- BITS : no ok
- In the case of a VHD, the Adm session does not open ( ? )

### Anomaly corrected with 1809 version

- The "USB ejection" icon (USB Eject Storage): finally a solution but with a huge delay
- MP4 ok in WMP
- BSOD : "software\...\bSideBySide" from the reference disrupts the start-up of "Services.exe". Randomly, I injected these keys from the ADK software hive and it works: no explanation!
- With an ISO file and when opening the ADM session, the warning "publisher not verified..." no longer displayed. The "smartscreenps.dll" file was missing

### Corrected anomalies with 20H2

- Correct installation of my scan ( key ... \control\class\{6bdd1fc6.. incomplete in Winpe)
- Changing/Checking the "FbWf.sys" change procedure and its signature
- The opening of the ADM session was disrupted: in the case of a boot on ISO, the "usrclass.dat" file must exist. This is not necessary in a Winpe Flat in VHD.
- « BITS » service= OK  
It requires the "brokerInfrastructure" service. But the latter should not start too early otherwise the taskbar icons are displayed with a delay of several minutes.

### Anomaly appeared with version 20H2

Connecting to a network sharing with the "explore" IHM sometimes generates a "RPC server" error. But the command "net use z:... " Works properly.

### Persistent or recurrent abnormalities

- Powershell's first very long launch: pb of .cat files

If I copy all the cat files from "Install.Wim" then it takes a minute before powershell becomes operational. With Procmon, we find that an Svchost creates the file «...\cartoot2\...\catdb».

- Can't pin in the taskbar with a right click
- Remote powershell et Wsman? Watch out for assemblies !
- Network trace with netsh: "ndiscap.sys" starts, the ETL file is generated but not the CAB file

# WinPe Knowledge Party



## The 3 pillars of a custom Winpe

These are the current basis for integrating Explorer's office into Winpe.

### ***The modification of the hives: "C:" replaced by "X:"***

Since at least the 10586 build, my scripts do not make this replacement. But with the 20H2 version, this change is again necessary.

### ***ProductPolicy***

The «HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Control\ProductOptions\ProductPolicy» key The toto key contains the information needed to activate certain features or components such as the wallpaper, the program «snippingtool.exe», etc.

Its content changes with Windows versions and with installed components.

This key is exported from an active Windows10 because I don't know how else to do it.

There's software on the net that allows you to visualize. It allows you to change this key to offline mode.

<http://reboot.pro/topic/20585-productpolicy-viewer/>

<http://www.remkoweijnen.nl/blog/2010/06/15/having-fun-with-windows-licensing/>

### ***Changing « runas »***

"Exploring" implements a set of COM elements. The COM dialogue uses customers and COM servers. The COM dialogue implements permissions according to the configuration of the COM elements defined in the register. For "Explorer," many COM components are configured to request permission. But Winpe does not implement a mechanism to respond to these requests for authorization.

To circumvent this security, several sites explain that it is necessary to change the values «HKCR\APPID...\Runas» for COM objects( all preferably ).

This point remains mysterious. But as I understand it, the absence of the "Runas" setting tells the COM server that it does not need to verify the identity of the logged user. If this setting is present (and equal to "InterActive"), the server controls the user's identity. However, this check fails with WinPe because the "System" account is not an ordinary account.

To learn more about security in COM:

[https://msdn.microsoft.com/en-us/library/ms682359\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms682359(v=vs.85).aspx)

[https://msdn.microsoft.com/en-us/library/ms680046\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms680046(v=vs.85).aspx)

## Important keys: list to be completed

This list is very limited at the moment because it is not always easy to find reliable information.

### Key "SYSTEM \SETUP"

CmdLine	REG_SZ	Program to launch (Winpeshl, pecmd.exe...)
SetupType	REG_DWORD	0=Do nothing, show login screen 1=Run CMDLine then REBOOT 2=Run CMDLine then show login screen Important: it is reset after winpe starts! It is essential in FullFlat and also in the mechanism of safeguarding hives (pseudo-persistence of hives)
SetupPhase	REG_DWORD	?
SystemSetupInProgress	REG_DWORD	1 = The behaviour of many programs is changing. Many features are then blocked!
SetupShutdownRequired	REG_DWORD	0=Shutdown Poweroff (Hard power off) 1=Shutdown Reboot 2=Shutdown NoReboot (Soft power off)
AllowStart	key	It replaces the "net start... ». The service registered under this key will be started automatically during the boot

Sources :

<https://social.technet.microsoft.com/Forums/windows/en-US/b942a34d-c4a7-489c-bb01-45dd65fa9b20/setuptype-and-cmdline-at-hkeylocalmachinesystemsetup?forum=itproxpsp>

### Key "MiniNt"

In HKEY\_LOCAL\_MACHINE\SYSTEM\ControlSet001\Control\MiniNT.

It is created by one of the programs that start Winpe.

It originates in the BCD file, with the value "Winpe = YES". When winPe's "loader" analyzes the BCD file and detects this value, it creates the MiniNt key.

It is therefore impossible to change this value while loading Winpe

### Key values «Control»

Key SYSTEM\CurrentControlSet\Control\

PeBootType	REG_SZ	created by WpeInit.exe, it identifies the type of media used to start Winpe - « Flat » : the startup took place from a VHD file (or hard drive) - « Ramdisk:OpticalDrive » : Start-up uses a Ramdisk (Boot.sdi). The start-up took place from a DVD (ISO file in a VM)
------------	--------	--

SystemStartOptions		It contains certain BCD values such as : TESTSIGNING MININT ( 'Winpe' dans le BCD ) BOOTLOG and other unknown values When loading, the "Fbwf.sys" driver looks for the chains MiniNT, EnabledOnAllSkus, WinpeCacheThreshold. But I don't know who's informing the last two.
--------------------	--	--

### **Des clés «System\...\Control\Session Manager**

BootExecute	Launches a prg in mode « native » ( autocheck...) When?
BootShell	no information (...bootim.exe)
SetupExecute	SetupCl.exe : since nt4 at least, changes the SID of the computer Poqexec.exe ... : makes updates after reboot

### **Keys «System\...\services\PartMgr»**

<b>System\...\Services\PartMgr</b>		
<a href="https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/Winpe-storage-area-network--san--policy">https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/Winpe-storage-area-network--san--policy</a>		
Parameters	SanPolicy	4 = masks the machine's local discs when Winpe is loaded. It is also used by "WinToGo." But you can mount the discs with "diskmgr.msc"

### **Keys «Software\...\Windows NT\Winpe»**

Instdrive no information	no information
CustomBackground	WallPaperHost.exe displays the image contained in this file But this program is not launched if "Explore.exe" manages the desktop
DisableRemovableStorageInit	Do not boot 1394 devices when searching for unattend.xml, saving time with "wpeInit.exe"

### **Keys «Software\...\Windows»**

\SOFTWARE\Microsoft\Windows\CurrentVersion\Policies\System

<a href="https://docs.microsoft.com/en-us/troubleshoot/mem/configmgr/no-mouse-cursor-during-osd-task-sequence">https://docs.microsoft.com/en-us/troubleshoot/mem/configmgr/no-mouse-cursor-during-osd-task-sequence</a>		
EnableCursorSuppression	REG_DWORD	0 = Mouse cursor is not suppressed 1 = Mouse cursor is suppressed The mouse cursor display avoids an all-black screen between the end of the driver load and the launch of "explore.exe" (for almost 1 minute)

## **Winpe's "unattend.xml" file**

<https://docs.microsoft.com/en-us/windows-hardware/customize/desktop/unattend/microsoft-windows-setup-display>

It allows "WpeInit.exe" to receive various commands and settings.

## **Change due to DSIM commands: to be completed**

- The « Dism.../set-ScratchSpace» command modify the value: «\system\...\services\fbwf\WinpeCacheThreshold»

## **What you need to know about the pilot installation**

To install a driver, you usually need 3 files.

- A .sys file: this is the driver's code
- A .inf file: it describes the actions to be taken by the OS for installation
- A .cat file: this is the signature of the previous 2 files

All drivers are signed in the 64bit version (quid in 32 bits). This signature is contained in a ".cat" file. The same ".cat" file may contain the signatures of several drivers.

Some drivers like HTTP. SYS do not have an inf file (although they are provided by MS).

A .inf file can refer to another file. inf" (loading another driver, including controls ...)

## **General Principle of Installation for Winpe**

The installation of a pilot takes place in two stages:

- Pre-loading :

Usual context: Winpe is inactive, the "Boot.Wim" file is mounted in a working directory

The driver's pre-loading involves copying files from WinPe's "store" (DriverStore and CatRoot directory) and modifying various hives including "Drivers"

The signature is checked as soon as it is pre-loaded.

With Dism:

The necessary files are identified, inf, sys, etc.

The necessary ".cat" file is identified with the « signtool.exe »

We put the "Boot.Wim" file in a directory

```
DISM.exe /Mount-Wim /WimFile:%ImageWimpe% /index:1 /MountDir:%Mount%
```

The driver is installed from the source containing the useful files

```
Dism /image:%Mount% /Add-Driver /Driver:C:\Winpe10\hdaudio.inf
```

Dism verifies file signature

Dism updates directories « DriverStore », « CatRoot » and the « Drivers » hive

- Installation :

Context : Active Winpe

After Winpe starts, the load can be:

- auto-automatic if the driver is running a device PNP
- manual, as for Hdaudio.sys. In this case, one can use « DrvLoad.exe ».
- carried out by another programme. "Wpeinit.exe" installs the drivers needed to build the local network

Note: Loading "Hdaudio.sys" automatically installs the "Hdaudiobus" driver.

The installation is carried out by the WinPe OS. Files are copied from the store « driverStore\filerepository\... » to the drivers' directory ( ...system32\drivers\ ).

### ***Principle of installing a driver with file « .inf » and « .cat »***

This is the most usual case.

The pre-loading was made with DISM during construction. A special event (PNP detection, Wpeinit.exe,...) initiates the finalization of the installation.

The device manager allows the manual installation of such devices.

### ***Principle of installing a driver without a .cat file***

Some "inbox" drivers do not have a reference to a .cat file in their .inf file. This is the case for PDF and XPS printer drivers, for example.

But these pilots have a signature. This signature is part of another file that forms the basis of the OS's signatures.

The only method I found was to use DISM. The active OS must then be Windows 10. Dism makes the pre-installation (import in the driverstore).

The installation in Active Winpe is to be dealt with on a case-by-case basis.

### ***Principle of installing a driver without a .inf file***

It has a ".cat" file shared with other devices/components. This ".cat" file is often already included in Winpe's basic supply.

Keys must be built or copied in the "...system\controlset\" ». See the script and installation of Http.sys, NativeWifiP, Vwififlt.

### ***How do I find the drivers' .cat file?***

Read in a Winbuilder script from Win10PeSe:

```
// The cat file can be found by using the signtool.exe from the Windows SDK 8.0, use :
« signtool verify /kp /v /a c:\windows\system32\drivers\monitor.sys »
```

MS provides the "signtool.exe" program in the SDK.

For example, to find the ".cat" file of "hdaudio.sys" in the active OS :

```
"c:\Program Files (x86)\Microsoft SDKs\Windows\v7.1A\Bin\signtool.exe" verify /kp /v /a c:\windows\system32\DriverStore\FileRepository\hdaudio.inf_amd64_dab2294dc8af0030\HdAudio.sys

Verifying: c:\windows\system32\DriverStore\FileRepository\hdaudio.inf_amd64_dab2294dc8af0030\HdAudio.sys

File is signed in catalog: C:\WINDOWS\system32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Client-Drivers-drivers-Package~31bf3856ad364e35~amd64~~10.0.10240.16384.cat

Hash of file (sha1): 4417D4DE1E79592F6B38315112935CC87DE46C53
```

### ***Check the logs***

Consultation of logs is paramount during investigations. So we need to locate the useful logs.

Pour Winpe Inactif et Dism.exe	Pour Winpe Actif
? :\windows\inf\setupapi.dev.offline.log	X :\windows\inf\setupapi.dev.log

### ***Track driver loading when starting Winpe***

Changing the Bcd as well :

```
Bcdedit /store ...\boot.bcd /set {default} bootlog Yes
```

The « x:\windows\NtBtLog.txt » file contains the list of loaded and unassigned drivers. But without explanation. Sometimes it's the beginning of a trail.

### ***Installing a manually modified driver***

See FBWF.SYS

# Features

**I would describe here the information that was important in my research.  
I am not proposing a list of files or keys to add here.  
See the script for these details.**

## The « Explorer.exe » desktop

To get a desk with some comfort when handling windows, you need to implement the Themes service and the DWM (Display Windows Manager) window manager. The windows are then displayed with square corners and the color change of the title bar signals the active window.

### *Not very beautiful but enough*

- Without "Dwm" and with the "explore" window ribbon, there appears a disturbing black stripe above the ribbon.

"DWM" requires the CoreMessagingRegistrar service for the 10240 build and also the WindowsTrustedRt driver for the 10586 build (see Win10PeSe). How did they find it?

- Without the service "Themes", I failed to display a wallpaper image after the launch of "Explore.exe". This service requires DWM.
- Color "title" bar for the active window:  
the next key seems sufficient : HKLM\...\DWM\ColorPrevalence = 1
- With an active Winpe, the "Default" hive in Regedit corresponds to the "config\default" file. It is used by the "System" account to become "HKCU." Various keys used by "explore" must be injected to achieve operation without anomalies.
- **Point to check... but never check because now I use the entire HIVE software of ISO , which avoids key search by key ...\Software\Microsoft\Windows\CurrentVersion\Explorer\UserSignedIn=1** to avoid, for example, the delay before the icons of the taskbar appear.

The display of the "WallPaper" is still a little obscure to me. See further!

### *The minimum to know about "Explore.exe"*

«Explorer.exe» makes two different features

- it creates a desktop for the user who logs in
- it allows you to explore the PC's file system

The second can be easily disposed of without implementing the first.

This second feature is available in an original Winpe. The "Explore.exe file is missing but the feature is active. To verify this, in an active original Winpe, open "Notepad," "file/open" menu and you get a file explorer that allows almost every action on the file system. Only its implementation is uncomfortable without an office.

The first feature, the desktop, requires the "Explore.exe" file. If you launch this program from a "Cmd" box and trace with "Procmon," you can find the missing items.

Then Winpe must automatically launch "Explore" at startup. To do this, we must add the key «...\**windows Nt\winlogon\shell = explorer.exe**». When "Explore.exe" is launched for the first time, it tests



the presence of this key and installs the user's desktop. On the second launch, it opens a window to explore the file system.

## ***The "display desktop" feature***

### **Description**

The source discovered late : <http://theoven.org/index.php?topic=1722.msg20025#msg20025>

I should have started there. But now, how do you know? Hence the interest of an effort to centralize documentation, although the changes are very rapid.

The activation of this function :

- with the keys "Windows" + D
- clicking in the bottom right corner (1 mm wide) between the clock and the edge of the screen
- using the pop-up menu of the taskbar

In Winpe :

The "Shell\_TrayWnd" window receives (WM\_XXXX) messages sent by various activation sources but does not process them.

A little investigation with SPY++

Spy++ ( it is indispensable and present in "visual studio community" which is free) shows the following messages received by «Shell\_TrayWnd» :

Msg 0x579 wParam 3 lParam 0

Msg 0x43F

In windows10: sending the 0x579 msg (wParam 3 lParam 0) is enough to switch the desktop display.

Idea: Try to understand why "Explorer" does not process these messages using windbg.

Correction in Winpese :

The "Wind.exe" program and the "MsgHook.dll" dll provide the expected treatment.

Simplified Algo of these files: basically, when receiving the right message in the hook, reduce or enlarge the main windows for desktop display or return to the previous display.

Note: How this program is launched Wind.exe/Myhook.dll in WinpeSe in 2016?

The «users\default\ntuser.dat» file is used by «system» session. It is also copied to be used by the adm session.

The runonce key in this file contains several values for launching programs.

And the "autorun-autorun.exe" value allows the "autorun.ini" file containing the to-throw commands, including "Wind.exe".



To play the video setup in PS :

```
Add-Type -AssemblyName system.windows.forms
[System.Windows.Forms.Screen]::AllScreens
```

### ***The "Open Command Here" feature***

Site describing the "Open Command Here" feature (see option 8 on page) :

<http://www.tenforums.com/tutorials/3288-command-prompt-open-windows-10-a.html>

For implementation in the pop-up menu without using the Shift key:

<http://www.askvg.com/enable-open-command-window-here-option-in-context-menu-in-windows-vista>

Delete values :

HKEY\_CLASSES\_ROOT\Directory\Background\shell\cmd\extended

HKEY\_CLASSES\_ROOT\Directory\shell\cmd\extended

### ***The "Copy as Path" feature***

You have to press the "shift" button and make a right click by pointing at the file or directory in an explorer's window.

I don't know how to make this option permanent -> Resolved: See the key in the script

### ***The contextual menus of the "desktop"***

Add NCPA, DEVMGMT, « device and printer » ... see the script.

### ***The "NEW" pop-up menu***

Since the build V1809: I had done a bad analysis.

Since 20H2.: I do not do any special treatment for this feature.

### ***Change the name of the icon "This PC"***

Ref : <https://www.computerperformance.co.uk/registry/hacks-display-computername/>

Changing the key:

HKEY\_CLASSES\_ROOT\CLSID\{20D04FE0-3AEA-1069-A2D8-08002B30309D}

LocalizedString Expand\_SZ «%Computername% %Username%»

So I quickly see the user's name, "System" or "administrator" on the screen and I know what the active session is.

### ***Function «CopyProgress»***

With all the Software keys from "Install.Wim" it remains to find the right files. The "chartv.dll" file

Email : noelblanc.Winpe@free.fr

was missing.

### **Other pop-up menus**

Not having the command "Win -X", I opted to add the most useful commands for me in the contextual menu by clicking on the desktop.

## **WinSxs**

This directory has a fundamental role for Windows updates. It contains absolutely all the files (except drivers) that Windows can update. There are no files in the "system32" directory. It only contains pointers on WinSxs files. They are hardlinks. DISM creates "hardlinks" when it installs a driver or component. This can be seen in the log file "Setupapi.dev.offline.log" for example.

Some of the files in WinSxs are compressed. This is the case for FOD files.

"SxsExp project" allows you to decompress such files.

## **Wow64 : 32-bit subsystem in Winpe**

Some versions of Winpe required the creation of 2 "system" objects. But now they are created by Winpe (see the archive at the end of the document).

Subsystem32 also requires the "WinSXS" and "SysWow64" directories.

## **MS EDGE With 20H2**

In the case of the use of Winpe in a VHD, Edge can be used by copying the directory in the VHD: "ISO\Program Files (x86)\Microsoft\Edge\Application."


Its size of more than 300MB remains surprising and important.


The download is operational with the "System" session (unlike IE64).

PDF files are read without further addition.

## **IE Full64 : almost complete (soon an archive)**

IE needs the key : HKEY\_CURRENT\_USER\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings

 if the hku\.default\...\cache\persistent (dw 1) value is absent, then IE launches 3 rundll32 : to delete user's activity.

 Note the presence of a test program :

"X:\Program Files\Internet Explorer\iediagcmd.exe".

It uses Dxdia,nesth,ipconfig... The product log file seems complete to me.



« The next key displays the last directory used to store a downloadable file from Internet Explorer, which could give a fairly good idea as to where the user stores his/her files. »

HKCU\Software\Microsoft\Internet Explorer\Download Directory

<https://www.forensicfocus.com/downloads/windows-registry-quick-reference.pdf>

## **clé x86AppPath**

3 tests :

- First test : The 32-bit subsystem is active
  - 64-bit iexplore launch: fugitive display
  - Copying of the x86 directory of internet explorer: a window appears and Procmon reports that iexplore-32-bit restarts non-stop.
  - iexplore-32-bit launch: "0xc0000034 impossible start" error message display.
- Second test: 32-bit subsystem inactive but the x86 directory of internet explorer is present
  - iexplore-64bits launch: The 64-bit version is trying to launch the 32-bit version. Viewing a window.
- Third test: I launch iexplore-32bits: display of a window but cursor "activity in progress"

First search with Procmon :

Iexplore-64bits consults the key hklm\software\microsoft\internet explorer\main\

**x86AppPath = x:\program files (X86)\internet explorer\iexplore.exe**

Then "createProcess" of this 32-bit version with the settings

«scodef:2760 credat:xxxx /prefech:2» où scodef fournit le Pid du process iexplore 64 bits.

- New tests:
  - remove this value: fugitive display
  - point "x86AppPath" to the 64-bit version: you get to see several 'IE' in the taskbar and view the content of 'MSN'.

## **Utilité de Loosely-Coupled IE (LCIE) ?**

First discovery on the internet

<http://blog.httpwatch.com/2009/04/07/seven-things-you-should-known-about-ie-8/>

"TabProcGrowth=0 is a registry edit that will disable the Loosely-Coupled IE (LCIE) function in Internet Explorer 8. Essentially what this means is that all tabs in Internet Explorer will be handled by one process of iexplore.exe. This also means that Protected Mode will be Off and that if one of your tabs crash, Internet Explorer will crash."

There are no more instances of "iexplore" with the "scodef" setting.

### ***The main IE window containing the menus is displayed!***

I did the test with the 1603 build 14393 version. Chance once again... All that was missing was the "IeFrame.dll.Mui" file.

### ***F12 ( debugger IE )***

At the moment it is not fully operational. The network track works.

### ***Anomaly identified***

Session «System» :

- Download: Files are well downloaded to a local directory but are not displayed
- Various menus don't work

Session «ADM» :

- Password management doesn't work
- Downloads work as well as viewing downloads

## **Powershell : very slow start**

Powershell's first start is always very long.

In the forum <http://theoven.org/index.php?topic=1639> : "sezz" explains :

I also figured out why starting PowerShell in Windows PE is extremely slow (first start takes about 20 seconds for me):

- NGEN hasn't run yet -> the native image cache doesn't exist
- CATDB hasn't been generated yet

Running "NGEN.EXE update /NoDependencies /Silent" takes ages, so I ran it once and fetched the files from "X:\Windows\assembly", now I theoretically could use them everytime when building an image, but they are very large (about 300MB)...

The CATDB gets created on the first start of PowerShell.exe, that's why it takes so long. The file "X:\WINDOWS\SYSTEM32\catroot2\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\catdb" can be backedup (stop CryptSVC service first) and then used when building an image. It's only about 20MB.

Couldn't find a solution yet without running Win PE first and grabbing the CATDB, but it's good enough for now :)

But I don't understand what it takes to go faster.

## Changing the name of the "computer"

Winpe is designed to use an "Unattend.xml" file. See the doc at MS.

Winpe normally generates a random name for the "computername."

There are 2 machine names, one for the machine and one for the network.

WinPeSe's script indicates that the next key must be changed to impose a predefined name :

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\WinPe
    SetComputerName = 0 type Reg_DWORD
```

Then the name of the "computer" must be entered in the following 3 keys:

```
...\SYSTEM\ControlSet001\Control\ComputerName\ComputerName
    ComputerName = «MyComputerName»
...\SYSTEM\ControlSet001\Control\services\TcpIp\Parameters
    HostName = «MyComputerName»
...\SYSTEM\ControlSet001\Control\services\TcpIp\Parameters
    «NV HostName» = «MyComputerName»
```

## Logging in with the "administrator" account

Without the work of the WinPeSe team, I would never have been able to implement login with the "administrator" account.

Native sessions in WinPe :

- Session 0 : for services
- Session 1 : automatically created by WinPe when it starts for the "system" account.
- Session 2 : it will be created when the session is opened by the "adm" account

In this "adm" session, the "administrator" account has all the rights since it belongs to the group of directors. But not all of these rights are active !!! Bad surprise!!!

### ***The general principle read in WinPeSe in Theoven.org***

When starting, Winpe activates the session for the "system" user.

To use the "ADM" session, the PC must have joined a workgroup.

The keys to the automatic opening of a session are prepared: "AutoAdminLogon mechanism."

It is absolutely necessary to stop and modify the "start demand" configuration of the two Gpsvc and TrustedInstall services.

Since the "system" session, we disconnect the console from this session with "tsdiscon.exe"

The "system" session is not closed. But the "keyboard/screen/resources" console is no longer assigned to this session.


Winpe understands that the "keyboard/screen/resources" console is back available. It therefore proposes to another user to log in with the "logon" GUI.

However, the "AutoAdminLogon" mechanism has been configured beforehand. So Winpe has the authentication elements of the new user (workgroup, account, password)

The login mechanism controls the identity of the "adm" account. Then create the profile directory for that user if it doesn't already exist.

There are therefore two interactive sessions, one on the "system" account and the other on behalf of the "administrator".

After this first switch, we switch from one to the other with the command "tscon," "tscon 1" to activate the "system" session and "tscon 2" to activate the "administrator" session.

 To avoid the empty password trap, I change the administrator's password.

### ***The modification of LSM.dll***

by NyaMisty in [github.com/NyaMisty/PELSMHooker](https://github.com/NyaMisty/PELSMHooker)

<http://theoven.org/index.php?topic=2881.msg34176#msg34176>


A real great work !

### ***The 'computer' must belong to a 'workgroup'***

It is necessary to ensure that the "computer" has indeed joined a "workgroup" so as not to have the error message "domain or workgroup inaccessible".

If you force the value of the 4 keys that define the computer's name, then the computer is not automatically registered in a "workgroup."

To join a "workgroup," you can use WMI's "joinDomainOrWorgroup" API, for example.

 Script in vbs to test in WinPeSe

```
strComputer = "."
Set oWmiService = GetObject("winmgmts:\\.\root\CimV2")
Set colComps = oWmiService.ExecQuery ( "Select * from Win32_ComputerSystem" )
For Each oComp in colComps
  wscript.echo "oComp.workgroup"
  res = oComp.JoinDomainOrWorkgroup("Workgroup")
  wscript.echo res
Next
```



Script in Ps for MicroWinpe.

```
$s = gwmi win32_computersystem
$s.JoinDomainOrWorkgroup('WorkgroupWinpe')
```

 Surprise :

With a right click on "properties" of "This PC" we see:

"Working Group: Not Available"

Then if you click "Change the Settings" and then "Computer Name" you see:

"Workgroup - 'Unknown'"

**?** What to check :

If the value of the 4 keys defining the name of the computer is not fixed, it seems that "Wpeinit" automatically integrates the computer into the "workgroup." You can also use a "unattend.xml" file to fill in the name "Computer Name." But what about the network?

### ***The administrator's profile directory: to review!***

Winpe's "Default" hive does not contain the "Environment" key. The "Winlogon.exe" program launches "Userinit.exe" and passes it a set of environmental variables including USERNAME.

Missing "Environment" key : USERNAME = SYSTEM

Key "Environment" present : USERNAME = ADMINISTRATOR (according to language)

The impact is important if you use a VHD and a remote session to connect with the administrator. Both sessions use the same desktop and/or a new directory is created with each administrator's connection in the USB drive. And so we lose the latest additions.

### ***Slowing down at opening: corrected***

There are many sources of slowing down of the 'administrator' login. I've identified these:

- "Patient..." message displayed for 10 minutes

This happens if both GpSvc and/or TrustedInstall services are not configured with 'start-demand'

- The network is set up before the login  **I don't see that point anymore**

It's wpeinit who builds the network. With procmon, I find that winlogon is trying to use the resource :

**\\LsaSetupDomain\*\MAILSLOT\NET\NETLOGON**

A time-out of about 6 seconds seems active. And the call is repeated several times.

I did two tests:

- removing the network card from the VM

- presence of the card but removing of the launch of the "Wpeinit" program

In both cases: the entries are not done because the maillot is "disconnected" and there is no slowdown.


- ntuser.dat ne contient pas "UserSignedIn" : à revoir !

The absence of the key <hkcu\...\explorer\UserSignedIn> introduces a delay of almost a minute after the display of "Preparation...".

- the 'SENS' service I set up for BITS introduced another two-minute delay. Winlogon sends notifications to various programs and awaits their response. In my case, the 'SENS' service was not working properly.

Winlogon notifications can be seen in this key that needs to be changed:

"...\ControlSet001\Control\Winlogon\Notifications\Components\"

 Surprise : "LsaSetupDomain" is also visible in SMB frames sent to the network when connecting to a printer, for example.

### **Error 0x80000003 : several cases since build 14393**

In the case of the opening of the "administrator" session, the error appears 0x80000003. It is not blocking. Just click the "OK" button and the login takes place.

In the case of the "switch user," the attempt to open a new session blocks the system.

In the case of a start from a VHD in "flat" mode (from a hyperV VM or from a USB drive), the session "administrator" does not open.

By chance, a site offered a 4GB file containing a version of Winpese and a set of installed applications (mainly portable applications that did not require searches for their integration) for download. See <http://theoven.org/index.php?topic=1873.0>. And now it's all working. This led to an effective solution.

Solution :

The ADK's "**Image.dll**" file is smaller than the ISO reference.

The "InputSwitch.dll file was also missing.

Other possible bypass (for memory):

rename Windows.UI.Logon.dll to Windows.UI.Logon.dll-org

### **Disabling the graphic part of logonUI**

It is possible to disable the graphic part of "logonUI." Just rename the file 'windows'. UI.logon.dll" (in "windows. Ui.logon.dll.old" for example).

The logonUI part in console mode ("ConsoleLogon.dll") displays a black box and shows the same texts but without the animations.

Another possibility: replace "logonUI.exe" with "cmd.exe"

This allowed me to track with "Procmon," although to no avail. But the method can serve me one day, maybe:

- Rename logonUi.exe to logonUi-org.exe
- Copy cmd.exe to logonUi.exe
- Prepare the keys and commands needed to log in with administrator
- Launch Procmon
- Start the login process : Tsdiscn
- CMD takes the hand. Launch a powershell.
- Recover the CMD control line :

```
gwmi win32_process -filter "Name='logonUi.exe'" | select -expand CommandLine
```

- Then launch "logonUi-org.exe" with the parameters of the "cmd" line.

When failed, "logonUi-org.exe" loop. We can do a "CTRL-C" to get out.

So we're going back to the CMD. And with ALT-TAB, we're back to powershell. We can then return to the "System" session with tscon 1. And arrest Procmon.

### ***Runas.exe only from the ADM session***

We can check the proper functioning as well :

```
net user MonToto MyPassword /add  
net localgroup administrateurs MonToto /add  
runas /noprofile /user:MonToto cmd.exe  
puis taper le mot de passe «MyPassword».
```

Don't forget the setting /noprofile» otherwise there is the error "the device is not ready".

Runas command doesn't work in System session.

### ***The warning "The editor could not be verified": corrected***

The "smartscreenps.dll" file was absent.

When Winpe starts from a USB stick or in a VM with an ISO file, all programs launched by "Windows-R" trigger the display of this warning. But if I unpack the "Boot.Wim" file in a VHD (with a Suitable BCD), this warning doesn't appear.

I find that the only program that does not trigger this warning is: "tscon.exe"

Bypass: When this warning appears, switch to the "system" session with "tscon.exe 1". Then go back in the session "adm" with "tscon 2".. The current powershell program and its descendants will still

trigger the warning but not the new programs.

With the "Open Command Here" feature, the launch of the "cmd.exe" program does not trigger this warning.

I've been looking for several years, I think.

## The network trail with netsh

[https://technet.microsoft.com/en-us/library/dd878517\(v=ws.10\).aspx#bkmk\\_traceStart](https://technet.microsoft.com/en-us/library/dd878517(v=ws.10).aspx#bkmk_traceStart)


Currently :

- it remains incomplete
- The PLA service is mandatory for a "pretty good" operation of the "stop" command.  
This PLA service requires changing the ACL key «service\pla\configuration»
- the "stop" command does not generate all the expected files (no file .cab...)
- The ETL file only fills up if the firewall allows for "inbound connections" or if an exception is created

The collection of System information requires the "Schedule" service. But it blocks the copying of files in a VM. So I'm not starting it. Variable depending on the version.

A lot of research for a small gain, the firewall log contains a better summary of the network frames.

The use of providers is currently performing better than using the scenarios.


 The "Upgrade Assistant" tool that I used to force the update to "Windows Creator" installs the "c:\Windows10Upgrad" directory on the hard drive. In this directory is the script "EnablerWifiTracing.cmd" which carries out the implementation of this trace for the WIFI network easily transposable for ethernet.

### ***The installation of the "ndscap.sys" driver : See new method***

Files used : ndscap.inf, ndscap.sys , « .cat » identified with 'signtool'.

The installation of the driver in the "network" battery is carried out after the start of "Winpe" :

```
netcfg -c p -i MS_NDISCAP
```

 The "-e" setting of "netcfg.exe" prohibits the installation of the driver. Incomprehensible!

It requires the file «ndscapfcg.dll». Start-up succeeds : net start ndscap

Capture is possible with a Wi-Fi or Ethernet card. Okay, in a VM.

### ***The configuration with netsh***

The keys "netdiagfx" and "Nettrace" that contain the available scenarios must be added to the

hive ."system.

Both keys are protected by ACL.

- To start a capture with a provider :

```
netsh trace start provider=Microsoft-Windows-TCPIP capture=yes report=yes correlation=yes overwrite=yes level=5
```

- To start a capture with a scenario :

```
netsh trace start scenario=InternetClient capture=yes report=yes
```

- To stop a capture :

```
netsh trace stop
```

The command "netsh trace stop" does not generate the file ".Cab" or the "report" file.

- To convert the file ETL to TXT :

```
netsh trace convert input=<chemin>\NetTrace.etl dump=TXT
```

The "txt" file does not contain all the expected information but TCP traffic is present.

See <https://isc.sans.edu//diary/No+Wireshark?%2bNo%2bTCPDump%3f%2bNo%2bProblem!/19409>

And see: <https://www.microsoft.com/en-us/download/details.aspx?id=44226>

## The event log

Don't forget to enrich the keys '...\services\eventlog\applications' et '...\services\eventlog\system'.

'Eventlog.msc' displays the newspapers well. To do this, the following sequence must be implemented :

- stop the eventlog service
- rename the key MiniNt
- restart eventlog service

With 20H2, I modify many ".dll" files to bypass key tests « MiniNT » and « SysemSetupInProgress ».

## Audits

The activation of certain audits enriches the "Security" event log.

```
auditpol /list /category /v
```

Category/Subcategory	GUID
Accès aux objets	{6997984A-797A-11D9-BED3-505054503030}
Accès DS	{6997984F-797A-11D9-BED3-505054503030}
Changement de stratégie	{6997984D-797A-11D9-BED3-505054503030}
Connexion de compte	{69979850-797A-11D9-BED3-505054503030}
Gestion des comptes	{6997984E-797A-11D9-BED3-505054503030}
Ouverture/Fermeture de session	{69979849-797A-11D9-BED3-505054503030}
Suivi détaillé	{6997984C-797A-11D9-BED3-505054503030}
Système	{69979848-797A-11D9-BED3-505054503030}
Utilisation de privilège	{6997984B-797A-11D9-BED3-505054503030}

To find out what security events are currently being traced :

```
auditpol.exe /get /category:*
```

The two commands to launch :

```
'auditpol.exe /set /Category:{6997984C-797A-11D9-BED3-505054503030} /success:enable /failure:enable'
```

```
'auditpol.exe /set /Category:{69979848-797A-11D9-BED3-505054503030} /success:enable /failure:enable'
```

The command that generates a Winpe BSOD (in 2016) :

```
'auditpol.exe /set /Category:{6997984A-797A-11D9-BED3-505054503030} /success:enable /failure:enable'
```

## The firewall: its logs, its configuration, profiles

Some commands used by "gatherNetworkInfo.vbs" to find out the current state of the firewall :

```
netsh advfirewall monitor show currentprofile
```

```
netsh advfirewall monitor show firewall
```

```
netsh advfirewall monitor show consec
```

```
netsh advfirewall firewall show rule name=all verbose
```

```
netsh advfirewall consec show rule name=all verbose
```

```
netsh advfirewall monitor show firewall rule name=all verbose
```

netsh advfirewall monitor show consec rule name=all

## **The logs**

With the audit, the event log records the activity of the firewall. Nevertheless, it is possible to get a more specialized log file. To do this, launch "WF.msc." Then click on "properties" (a small link under the last profile), and activate the logs.

☛ Warning: Entries in the log file are at least 30 seconds late on the event. So patience!

## **Activation/deactivation**

It seems to me preferable to leave the service active and allow all flows. Many apps query the "Firewall" and don't work if its APIs don't respond. Same entry point for the setup as above.

For example, if you stop the firewall service with "net stop MpsSvc," then the "ping" to WinPe fails. But if you start the firewall with "net start MpsSvc" and allow "inbound connections" with "Wf.msc," then the "ping" to WinPe succeeds.

Useful with WinPe: Wpeutil disablefirewall ou Wpeutil enablefirewall

## **How do I change the profile of the public network in private? NOT OK!**

???? Too old ????

Information found on the internet using the "NetProfM" service (Network List Service) and its DCOM interface (see the GUID of the order):

```
# Get network connections

$networkListManager =
[Activator]::CreateInstance([Type]::GetTypeFromCLSID([Guid]"{DCB00C01-570F-4A9B-8D69-199FDBA5723B}"))

$connections = $networkListManager.GetNetworkConnections()

# Set network location to Private for all networks

$connections | % {$_.GetNetwork().SetCategory(1)}
```

The Network and Sharing Centre shows the change. But WF. MSC indicates that the profile is still public, so this change is inoperative.

Other sites

With GUI's 'home folder' :

<https://tinkertry.com/how-to-change-windows-10-network-type-from-public-to-private#Fix2WiFi>

With the register :

HKEY\_LOCAL\_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles

Email : noelblanc.Winpe@free.fr

puis dans les sous-clés, la valeur «Category» décrit le type de profil :

Public: (leave this blank)          Private: 1          Domain: 2

With Powershell

<http://www.tenforums.com/tutorials/6815-network-location-set-private-public-windows-10-a.html?ltr=N>

Get-NetConnectionProfile

Set-NetConnectionProfile -Name "xxxxxxxxxx" -NetworkCategory Private

a little information about nla and firewall:

<https://blogs.technet.microsoft.com/networking/2010/09/08/network-location-awareness-nla-and-how-it-relates-to-windows-firewall-profiles/>

## The "Network and Sharing" centre: with build 14393?

The NetProfM service needs to be installed. And to make this service operational, the key "SystemSetupInProgress" must be temporarily neutralized while it starts

Not good, need corrections !!!

## WinRm in the adm session: OK in both directions

### *Summary description*

The WinRm service (previously called WsMan in Windows 8) allows YOU to send PS commands to a remote machine. It also allows you to receive them.

It uses two ports, one (47001) to set it up, the other (5985) to receive commands launched from a remote computer.

After starting it, we can verify that it has opened the port TCP 47001 with the order

netstat to get the pid (or name with -anb) and a loop consultation if needed!

```
NetStat -ano
TCP 0.0.0.0:47001    0.0.0.0:0    LISTENING    4
TCP 0.0.0.0:5985    0.0.0.0:0    LISTENING    4
```

WinRm requires a setup step with the "Winrm quickconfig -q" command. Then, outside the domain, you have to declare the machines of trust.

```
Winrm Quickconfig -q
Winrm Set winrm/config/client '@{TrustedHosts="*"}'
```

But as it stands, these commands fail.



Anomaly found: various messages "access denied"

What can I do? change the WinRm account to 'LocalSystem' or change the content of local groups as follows:

```
net localgroup WinRMRemoteWMIUsers__ /add
net localgroup administrateurs system /add
net localgroup administrateurs localservice /add
net localgroup administrateurs networkservice /add
```

Is there a "winrm" account in ACL?

I always have the same "access denied" error. And then a stroke of luck that falls under the serendipity (a question of the game of a thousand euros in France).

Discovered by chance, with the start of the service "File Server", we advance a little and we get the error already encountered in Winpe4 or 5 for Windows 8.1.

```
Net start lanmanserver
```

I also change the password to the "administrator" account so that I can establish "incoming" remote connections. I share a resource for possible copies of files. And I'm installing the "SecLogon" service.

```
net user administrateur +Noel
net share monx=x:\
```

The new message for "Winrm QuickConfig -q" becomes:

```
WSManFault
  Message
    ProviderFault
      WSMANFault
        Message = WinRM firewall exception will not work since one of the network connection
types on this machine is set to Public. Change the network connection type to either Domain or
Private and try again.
```

This is a warning that the TCP 5985 port is not open for WinRm incoming traffic.

The configuration of trusted machines succeeds:

```
Winrm Set winrm/config/client '@{TrustedHosts="*"}'
```

➤ For Outgoing WinRm commands

Powershell does a Winpe environmental check and reports that remote access with WinRm doesn't work in Winpe. We can temporarily neutralize the MiniNt key and launch Powershell. And so the outgoing WinRm commands succeed.

Email : noelblanc.Winpe@free.fr

To temporarily neutralize the MiniNt key:

```
reg DELETE HKLM\system\currentcontrolset\control\miniNt /f
start-process PowerShell -argumentList '-executionpolicy unrestricted'
start-sleep -s 1
reg ADD HKLM\system\currentcontrolset\control\miniNt /f
```

To make sure WinRm outgoing commands are working:

```
$s="192.168.0.12" # my win10 computer
# ask the password for the remote computer
$c=get-credential WIN-LBH1HBGLMAA\noel
invoke-command -computername $s -credential $c -scriptblock {$env:computername}
invoke-command -computername $s -credential $c -scriptblock {gwmi win32_bios}
```

- For WinRm incoming commands

I open the port with netsh.

```
netsh advfirewall firewall Add rule name="NONO-5985-winrm" dir=in protocol=tcp localport=5985 action=allow
```



Before V1809: to open the port, you have to start the MpsSvc firewall service. It seems that it is "wpeinit.exe" that starts it. And with the Adm session the script no longer launches this command at the right time. I change my PS scripts to open this port.

For testing, I manually launch the following commands from a console:

```
net stop mpssvc
winrm quickconfig
net start mpssvc
```

And so port 5985 is well "listening" in "netstat-ano".

The "test-wsman <ip of Winpe>" command succeeds when launched from a remote pc windows10.

Question: Do we really have to go from "public" to "private"?

<http://www.tenforums.com/tutorials/6815-network-location-set-private-public-windows-10-a.html?ltr=N>

From a remote pc, I launch the following commands to the "Winpe" machine:

```
$s="192.168.0.15" # my Winpe computer
# ask the password for the Winpe computer
$c=get-credential MINWINPC\administrateur
invoke-command -computername $s -credential $c -scriptblock {$env:computername}
```

I get the message "[192.168.0.15] The connection to the remote server 192.168.0.15 failed with the following error message: The WSMAN service could not initiate a host process to process the given request. Make sure the vendor's host server and the WSMAN proxy are properly registered."

The Procmon track shows that the Winrm service does not launch the "wsmprovhost.exe" program.

I change the following key in the Winpe machine:

**hklm\system\setup\SystemSetupInProgress = 0**

Then I restart the "invoke-command" command from the Windows 10 machine.

And the order succeeds!

### ***With 20H2***

In the first tests with 20H2, the WinRm configuration was in check. During the search, I changed the sequence of "post-start" operations. Now, the "join a workgroup" action takes place before WinRm is set up. And WinRm is working properly.

## **Wuau servicing : update**

It can be used to make auto-updates of drivers.

It uses BITS.

TO DO

## **BITS**

It makes it easy to download.

The BITS service requires 'assemblies' and very few files (mainly qmgr.dll).

I encountered various and different difficulties in the son of Winpe's versions.

The last was the "aassembly" file copy whose length of the path exceeded the 256-character limit. PS commandlets do not handle such items.

"Bitsadmin.exe" or Powershell can be used.

### ***With 20H2***

The "Import-Module BitsTransfer" command was in check. Assemblies were missing. PS's copy-item command does not support names over 256 characters and an "assemblies" was not copied. I now use "robocopy" for some copies.

The BITS service requires the BrokerInfrastructure service. But if the "BrokerInfrastructure" service starts before "Explore," then the icons will appear with a delay of about 5 minutes. So I do "net start BrokerInfrastructure" in a start script launched by "WpeInit".

## Remote desktop: MSTSC and TermService

First findings :

- Winpe's "outgoing" sense :

The error message is: "The remote computer requires Network Level Authentication which your computer does not support."

- Winpe's "ingoing" sense :

"impossible connection" even after allowing "inbound connections" in the firewall.

The "TermService" service signals that it is starting. But in the "Windows/TerminalServices-localSesionManager" event log, a message says "remote office services do not accept login because the installation program is running."

The "TermService" service does not listen to port 3389. Finding made with "netstat-ano."

### ***MSTSC from Windows10 to WinPe : it's possible***

I do not describe the list of keys, files, drivers and services. Please see the script «Traitement.ps1».

### ***MSTSC from WinPe to Windows10: it's possible***

#### **GUI de logon**

When you launch MSTSC from Winpe, a black box appears and offers to enter the ID and password. To find the graphic connection interface, you need to rename the MiniNt key.

#### **NLA**

<https://technet.microsoft.com/en-us/library/cc732713.aspx>

**Network Level Authentication** is an authentication method that can be used to enhance RD Session Host server security by requiring that the user be authenticated to the RD Session Host server before a session is created.

**To determine whether a computer** is running a version of Remote Desktop Connection that **supports Network Level Authentication**, start Remote Desktop Connection, click the icon in the upper-left corner of the Remote Desktop Connection dialog box, and then click About. Look for the phrase Network Level Authentication supported in the About Remote Desktop Connection dialog box.

<http://vidmar.net/weblog/archive/2007/05/27/The-remote-computer-requires-Network-Level-Authentication-which-your-computer.aspx>

#### **Pre-verification of MSTSC's NLA configuration on WinPe**

- Launch MSTSC

Email : noelblanc.Winpe@free.fr

- left click on the icon in the top left corner or right click in the title bar
- menu «About»

A window opens to indicate if NLA is supported. By default, NLA is not supported in WinPe.

## NLA deactivation on Windows10 machine

There is at least one configuration that works: disable NLA authentication on the remote windows10 machine.

On the Windows10 machine, part of the "system-use remote use" menu, the box "Only allow computers running the desktop remotely with NLA authentication" must be unchecked.

This allows you to connect with the remote windows10 machine.

## Activation of NLA on WinPe version 1607 build 14393

But is it possible to set up NLA authentication on WinPe? Yes!

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\Lsa\OSConfig

Security Packages Reg\_Multi\_Sz kerberos, msv1\_0 , tspkg, pku2u, cloudAP, wdigest, schannel

HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders  
SecurityProviders = credssp.dll

new files : tspkg.dll, credsp.dll

## Activation of NLA for WinPe version 1703 build ???

A little change with this version:?

- export/import a key

...\ControlSet001\Control\lsa\CredSSP

- create a key/value

key	value	data
...\ControlSet001\Control\SecurityProviders	SecurityProviders	credssp.dll

## AllowEncryptionOracle key

See : <https://support.microsoft.com/en-us/help/4093492/credssp-updates-for-cve-2018-0886-march-13-2018>

## Samba

Adapt the key «LSA\LmCompatibilityLevel»to your need

<https://docs.microsoft.com/en-us/windows/security/threat-protection/security-policy-settings/network-security-lan-manager-authentication-level>

## MSRA «remote assistance»

Just add a few files to the set already brought by MSTSC and RDP.

"The Expert" works in both sessions, SYSTEM and ADM.

But the "novice" must use the ADM session in Winpe.

## Quickaccess

This feature requires little investigative work. Again, the "Administrator" session should be used to offer help.

## IC for HyperV

Again, no major difficulty, read the script to know the list of items to use.

## Windows Defender Offline : little-known feature

The site <http://reboot.pro/topic/20497-run-windows-defender-offline-from-Winpe/> explains how to implement this feature. The tricky point remains the updating of the signatures

To test in Winpe :

- check out the site <https://support.microsoft.com/fr-fr/help/17466/windows-defender-offline-help-protect-my-pc>
- download the 32 or 64-bit file as needed ( par exemple <https://download.microsoft.com/download/1/E/D/1ED15A90-4014-491D-9C96-DEA70B99DD5E/mssstool64.exe> )
- launch this program « mssstool64.exe »
- Choose the "create an iso" option
- mount the iso « C:\ProgramData\Microsoft\Microsoft Standalone System Sweeper Tool\WDO\_Media64.iso » in G: drive for exemple
- in the root of the iso, take the 2 files
- in the "Boot.Wim" file, retrieve the directory « G:\sources\Boot.Wim\Program Files\Microsoft Security Client\ »

## Installing hdaudio drivers: to review

Installing a manufacturer's audio drivers increases the size of "Boot.Wim" and thus Winpe's loading time. These specific drivers do not always seem necessary.

It seems that the "hdaudio.sys" driver is enough in many cases to get the sounds in Winpe. Its installation requires some comments.

- The "hdaudio.inf" file has no reference to a .cat file, so "drvload" cannot load it under Winpe
- The simple copy of the .cat in the "catroot" directory is not enough for its installation: the hive "drivers" must also contain a reference to this driver.
- A quick test: take the hive "driver" of "Install.Wim" (but it does not contain the driver "fbfw" to write in X :)
- Can we pre-install it with Dism?

Yes with inactive Winpe : `Dism /image:%Mount% /Add-Driver /Driver:C:\Winpe10\hdaudio.inf`

Not with Active Winpe: the "online" mode is not supported by Dism for Winpe!

But you can rename the key « MiniNt » and Dism will become good

- Can we then load it with DrvLoad with Active Winpe? Yes
- Do you need a few more orders after Winpe starts? Yes.
- The codecs are visible with Msinfo.exe
- The MMCSS pilot. Is SYS mandatory? Not in my context!

Not all "audio" formats are recognized. For example, an MP4 file is not read. It depends on Winpe's versions

### ***Additional post-start commands : need TODO, see script***

I made the choice to pre-load the hdaudio driver.sys. With another installation method (close to WinPeSe) the installation of ACL would not be necessary.

The next sequence is required after Winpe starts.

- Starting the AudioEndpointBuilder service
- hdaudio driver loading :`drvload d:\sourceDesAjouts\audio\hdaudio\hdaudio.inf`
- changing key acles : 'HKLM:\SOFTWARE\MICROSOFT\Windows\CurrentVersion\MMDevices\Audio\Render'

This key is created after the AudioEndpointBuilder service is started.

- start-up of the audiosrv service
- COM recording for wmpplayer (useless with the full "software" hive )

## **WMPPLAYER : 32bits and 64bits inseparable**

My observation :

In the Software hive, file extensions are associated with WMP32bits

Test to do: point a file extension "Wav" for example to WMP64bits

Currently there are two possibilities to use WMP:

➤ Use only WMP 64

copy the WMP64bits directory in "Program Files (x86)"

Advantage: space saving

downside: not all formats are recognized, MP3 ok

implementation tested with v1709 :

Open WMP64. And DragAndDrop a MP3 file.

Anomaly: Double-click on a file generates an error after a delay of 1 to 2 minutes:  
"Server execution failed"

The behavior varies depending on the evolution of my tests and my additions.

➤ Use WMP 64 and WMP 32 as in "normal" windows 10

copy the WMP 32 directory in "Program Files (x86)"

copy the WMP 64 directory in "Program Files"

disadvantage: the size of "Boot.Wim" increases.

WAV, MP3, MP4 = OK

🔍 Anomaly with the System account:

The double-click on an MP3,WAV or MP4 file deposited on the desktop opens WMP but no sound (or image) because WMP searches for the file in the following location:

X:\Windows\SysWOW64\config\systemprofile\Desktop\Ring05.wav

### ***Old abandoned method***

Having not taken the time to change some keys, I copied the 64bits directory of Wmplayer from the source in the directory "program files (X86)". Indeed, it is in this directory that the OS will look for the WmPlayer program during a double click on an "audio" file.

Weird but I don't take the time to look for a better solution.

Note: with build 14393, it is impossible to read the files. Wav. But you can read the MP3 files.

### ***New method: WMP32 and WMP64***

MP3, WAV and MP4 files are read correctly.

### **API mciSendString of WinMm**

There are various ways to listen to music. This API is easily implemented in a PS script with a bit of C#.



I added in "Boot.Wim" some files and a key in "drivers32" (see script if needed).

See these links :

[https://msdn.microsoft.com/en-us/library/windows/desktop/dd757161\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd757161(v=vs.85).aspx)

for error numbers: [https://msdn.microsoft.com/en-us/library/aa228215\(v=vs.60\).aspx](https://msdn.microsoft.com/en-us/library/aa228215(v=vs.60).aspx)

for the signature c# : <http://pinvoke.net/default.aspx/winmm.mciSendString>

Different sites report an anomaly with the "open" command and the "mpegvideo" type used for MP3. But after a few tries, the PS script allows you to read Wav and MP3 files. This is even if the name of the file contains spaces.

Example in PS :

```
#
# permet de lire des MP3 et aussi des WAV dans WINPE
#
param([string] $fileAudio="")

Add-Type -Name mci -Namespace media -MemberDefinition @"
    [DllImport("winmm.dll", EntryPoint = "mciSendStringW", CharSet = CharSet.Unicode,
SetLastError = true, ExactSpelling = true)]
    public static extern int mciSendString(string lpstrCommand, string lpstrReturnString, Int32
uReturnLength, IntPtr hwndCallback);
"@

if ( $fileAudio -eq "" -or -not(test-path $fileAudio) ){
    "fichier absent : $fileAudio"
    return
}

switch ((([System.io.path]::getextension($fileAudio)).toupper())
{
    ".WAV"      {
                $type="waveaudio"
                break
            }
    ".MP3"      {
                $type="mpegvideo"
                break
            }
    default {
                "extension non traitée"
                return
            }
}

# ok with spaces in the name of file
$stringOpen = "open " + ""$fileAudio`"" + " Type $type Alias playsound" ;
$stringOpen=$stringOpen
$mediaError = [media.mci]::mciSendString($stringOpen, "", 0, 0);
"mediaError=$mediaError"
```

```

$stringPlay = "play playsound wait"
$stringPlay=$stringPlay"
$mediaError = [media.mci]::mciSendString($stringPlay, "", 0, 0)
"mediaError=$mediaError"
$stringClose = "close playsound wait"
$stringPlay=$stringClose"
$mediaError = [media.mci]::mciSendString($stringClose, "", 0, 0)
"mediaError=$mediaError"

```

## Wifi in WinRe

As we often read, Winre can be used as a basis for customizing his "Winpe".

The big advantage is that MS has already incorporated many features into this "Winre".

This is the case with WIFI.

## Wifi in Winpe

New (6 Mars 2018) : <http://www.sconfigmgr.com/2018/03/06/build-a-Winpe-with-wireless-support/>

### **My references**

<http://ploadletter.co.uk/2011/12/03/windows-pe-builder-script-for-waik-including-wifi-support/>

<http://www.serverwatch.com/server-tutorials/using-netsh-commands-for-wi-fi-management-in-windows-8.html>

Pour décrypter :

<http://stackoverflow.com/questions/10765860/decrypt-wep-wlan-profile-key-using-cryptunprotectdata>

<http://superuser.com/questions/133097/netsh-wlan-add-profile-not-importing-encrypted-passphrase>

Autres sites:

[http://blogs.technet.com/cfs-filessystemfile.ashx/\\_\\_\\_key/telligent-evolution-components-attachments/01-6127-00-00-03-31-62-58/Windows-7-Deployment-Procedures-in-802-1X-Wired-Networks.pdf](http://blogs.technet.com/cfs-filessystemfile.ashx/___key/telligent-evolution-components-attachments/01-6127-00-00-03-31-62-58/Windows-7-Deployment-Procedures-in-802-1X-Wired-Networks.pdf)

<http://www.msfn.org/board/topic/162453-Winpe-40-enable-wireless-support/>

### **What do we need to install?**

For Wifi, there are several components to install:

- drivers of maps provided by OEMs
  - I chose to do a pre-loading with DISM
- drivers used in the "network" layer
  - NativeWifiP

- Vwifflt
  - VwifiBus : It is installed automatically by the include netwifibus.inf
- network layer services
- MS\_NativeWifiP
  - MS\_vwifi

I chose to pre-load with DISM. And they will be installed with netcfg.exe.

With 20H2, I use a new method to avoid netcfg.exe. See below.

### ***Identify and collect files inf, sys, cat ...***

- NetServices for Netcfg.exe
- Netnwifi.inf for MS\_NativeWifiP service
  - Netwififlt.inf for MS\_Vwifi service
- The driver of my wifi card (intel in my case)
- NetWew00.inf, NetWew00.sys, NetWfw00.dat

The .inf file of the driver on my card has a 'include' :

See the log file«...\windows\inf\setupapi.offline.log »

- For 'include' :
- NetWifiBus.inf, VwifiBus.sys, VwifiBus.sys.mui (fr-FR ou En-us if beta)
    - \* The "dism /add-driver" command will pick up the driver in the directory :
   
     %mount%\windows\inf\vwifibus.sys
   
 So copy beforehand the file 'NetWifiBus.inf' in this directory.

### ***Copy the L2Schemas directory***

This directory is available in the ISO reference.

### ***Basic NetSh Commands***

It is sometimes useful to export your Wi-Fi profiles from your windows10 and import them into your Winpe. In a Win10 machine, set up Wi-Fi connections. Then, export these configurations in Xml files. They will contain SSDIs and passwords.

After starting WinPe, you'll need to import the desired profile. Then connect..

- Export of registered profiles with clear password in the current directory:

```
netsh wlan export profile key=clear
```

- Exporting a profile:

```
netsh wlan export profile name="freebox_opfm" folder="%SourceDriversWifi%\Profils" key=clear
```

- Importing a profile in Winpe:

```
netsh wlan add profile filename="x:\MesProfilesWifi\freebox_opfm"
```

- Wifi connection with the profile imported in Winpe:

```
netsh wlan connect name="freebox_opfm" ssid="freebox_opfm"
```

## ***A Wi-Fi script in PS***

On the CodePlex <http://managedwifi.codeplex.com/> site, I found a C# that allows Wi-Fi. I placed it in a PS script. I added a graphical interface as well as the disconnection function.

## **Nouvelle méthode pour préparer le réseau**

There are too many "netcfg" commands. I tried to copy WinRE's method.

By copying the "NetWork" and "WetWorkSetup2" keys, these "netcfg" commands can be avoided.

You have to take them on a windows10 or in WinRe.

## **Microphone**

The microphone now uses the "CamSvc" service.

## **Bluetooth**

The search was a bit long but without much difficulty.

Peering: The slowness of "DevicePairingWizard.exe" had led me to write my own "pairing" tool much faster.

"BHT-LE" devices do not work because there is no solution to perform "pairing." I seem to have read that this is an MS choice for Windows.

## **Printers in Winpe**

Printers are very different from each other (multi-function or not, laser or inkjet, etc.). They can be connected in various ways, network, usb, wifi, web... The architecture of the drivers has evolved in the thread of versions of the OS (kernel mode, user mode ...).

Finding the right driver is a tricky point. It is easier to let the system find the drivers on its own. This is what is done with printers shared on a server. When the printer is installed, the drivers in the server are

automatically downloaded to the workstation.

In the case of the installation of a USB printer, winpe will have to make the necessary drivers available either by embedding them in the "Boot.Wim" file or in another medium, USB for example.

I only have one printer, which limits my investigations. I can share it on the network or connect it in USB.

➤ Architectural reference site : <https://technet.microsoft.com/en-us/library/cc976755.aspx>

➤ to launch the printer addition GUI, type:

```
«X:\windows\System32\rundll32.exe" printui.dll,PrintUIEntryDPIAware /il»
```

➤ check the log file : «x:\windows\system32\inf\setupapi.dev.log»

### ***Invisible printers in devmgmt.msc: corrected***

The printers are clearly visible in notepad (print menu) but they don't appear in the configuration panel: I don't know what the solution was!

They are visible with powershell, "Get-printer" and "gmi Win32\_printer" after adding "mof" files.

### ***Invisible printers in "devices and printers": corrected***

Printers become visible after 2 minutes after the start of the spooler and DSMSVC services. It's a hard-working delay in a spooler program.

### ***My printer's special features***

This is the "samsung SCX-4500 Series" printer. However, drivers in the 64-bit version use 32-bit programs. So WOW64 is mandatory in this case.

### ***Network printer in the "administrator" session: it works***

With DISM, I installed the "InBox components" NTPRINT.INF and USBPRINT.INF.

After copying files and registry keys, the "spooler" service starts.

From the "configuration panel/devices and printers" we can start the installation of the network printer. It is necessary to provide the "credential" of a user authorized to manage the printer and the print queue. Check the various settings for sharing the printer on the server.

### ***Network printer in the "system" session: impossible***

The installation starts normally. The big .cab file is well downloaded. But finalization (text "finalization" appearing in the installation window) triggers the display "impossible connection to the printer: error 0x00000534.

The analysis of the procmon trace and the comparison with an installation on another PC confirm the download of the drivers and the discrepancy when LSA consults the SAM. The program

"PrintIsolationHost.exe is not launched in the case of the error.

The activity of "Spoolsv.exe" at that time:

- entering the account name and password of the user created on the server
- creating the registry key «\software\microsoft\windowsNt\CurrentVesion\print\providers\client Side Rendering Print Providers\servers\...\Client Side Port\...».
- downloading the .cab file containing the drivers
- "Finalization..." display (not always visible)
- deleting the previously created key
- Displaying the error message «Error 0x00000534»

When I searched the MSDN error codes site, I found this:

«error 0x00000534: No mapping between account name and SID was done».

Why the control? Maybe because a printer is an attribute of an account (a bit like in the AD). Test with «SystemSetupInProgress» ???

It's impossible for me to go any further.

Msdn website describing errors : <https://msdn.microsoft.com/en-us/library/cc231199.aspx>

### ***USB printer in the "administrator" session***

By installing this USB printer on another pc, and by analyzing the file "inf-setupdev.log", we find the directory containing the right drivers in the "driverstore".

This then allows you to try installing on Winpe, either with the configuration panel or from the device manager by choosing the printer and not the scanner in my case.

The installation is successful. The printer is visible in 'notepad/print menu'.

But the impression failed with the message "A call to StartDocPrinter was not made".

The procmon trace shows that the spooler (spoolsv.exe) is testing the key "SystemSetupInProgress".

Bypass:

- SystemSetupInProgress = 0
- start-up of the spooler service ( ? printing launch at least once before StartDocPrinter)
- SystemSetupInProgress = 1

Printing is operational. But here too, the printer is not visible in the "configuration panel"/devices and printers».

It's easier to install the printer from the device manager after you've connected it (avoids creating a port and a ghost printer).

The USB printer installed in one session is visible and available in the other session.

## **USB printer in the "system" session**

The installation and printing work as in the case of the "administrator" session.

## **Printer and "remote desktop"**

I'm throwing Winpe at a remote machine. And I use "remote desktop" (mstsc) to connect to this Winpe. My local printers are added to Winpe's local printers.

Again, the refresh in "devices and printers" is long and takes between 3 to 5 minutes. TO BE VERIFIED

## **The scanner of my "sumsung"**

For a long time I did not try to correct the following point. But I ended up finding the source of the problem.

The installation of the pilot of my scanner was proceeding correctly. But the "class install" part was not processed. The "Sti\_ci.dll" dll was not requested because Winpe's basic supply does not include all components in the system hive. The addition of the key ".class\{6bdd...}\000\install32" solves the anomaly. But it was easy to find.

☞ 🖨️ 🖨️ 🖨️ 🖨️ 🖨️ 🖨️ 🖨️ 🖨️ 🖨️ 🖨️ :

<https://msdn.microsoft.com/en-us/windows/hardware/drivers/image/wia-architecture-overview>

<https://msdn.microsoft.com/en-us/windows/hardware/drivers/image/wia-core-components>

☞ Activating WIA traces provides useful information for development

The log file: \debug\wiadebug.log.

Changing keys:

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StillImage\Trace\wiaservc.dll]
```

```
"TraceFlags"=dword:00000007
```

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\StillImage\Trace\sti.dll]
```

```
"TraceFlags"=dword:00000007
```

The defaults with the 20H2 are : 0x407

## **For the record only: Compulsory manual treatment**

At the end of the installation of the printer and scanner, some keys are not informed. It is therefore necessary, as it stands, to add the missing keys and to stop/reboot the "stisvc" service.

Changing keys:

```
HKLM\SYSTEM\CurrentControlSet\Control\Class\{6bdd1fc6-810f-11d0-bec7-08002be2092f}\0000
"SubClass"="StillImage"
"friendlyname"="My scan Samsung"
```

```
"DeviceID"="{6bdd1fc6-810f-11d0-bec7-08002be2092f}\\0000"
"capabilities"=dword:00000031
"deviceType"=dword:00000001
"deviceSubType"=dword:00000001
```

## ***The spooler service***

The spooler service is the centerpiece of printer architecture. Two key points must be considered.

- Identify the useful elements to install the spooler

This first step is to start the spooler without error.

Its installation is quite simple and requires little investigative work.

- Activate the spooler

Like many services (EventLog, TermService...) its start-up is not enough to make it operational. Winpe is a tool to install Windows.

Many features are not compatible with its efficiency, speed, robustness, etc.

The spooler is inactive if `hkml:\system\setup\systemsetupInProgress = 1`

- Changing security (ACL) de `..\spool\printers`

## ***PDF and XPS printers and the spooler***

When you have successfully installed the spooler, start it error-free, it is active, then you have to install printers.

Currently, it is the Spooler itself that installs PDF and XPS printers.

It does so 2 minutes after starting (as in a normal windows 10)

But it takes 10 minutes for printers to appear in device and printer.

A file change is required to remove these delays.

Note 1: I added 2 .mof files for Powershell

```
# pour gmwi win32_printer
x:\windows\system32\wbem\mofcomp.exe x:\windows\system32\wbem\win32_printer.mof

# pour get-printer
x:\windows\system32\wbem\mofcomp.exe x:\windows\system32\wbem\
PrintManagementProvider.mof
```

Note 2: The 'xpsrchvw.exe' file is now missing from the Windows PRO supply.

## **Some reminders:**

The printers in the adm session may be different from the printers in the system session.

Email : [noelblanc.Winpe@free.fr](mailto:noelblanc.Winpe@free.fr)



Printers redirected from a "remote desktop" session are added to local printers.

The refresh of "devices and Printers" occurs 3 to 5 minutes after the connection (to be checked).

In the system session, you can't install a printer shared by another microcomputer.

### ***New choice: changing files***

I modify several files to eliminate the various delays and tests of SystemSetupInProgress.

## **Ejecting USB devices**



Feature still random

My test context: boot on USB drive and vhd containing Winpe in Flat mode

The notification area at the bottom right displays an icon that allows connected USB devices to be ejected. However, the line identifying the device was not displayed. Which didn't allow him to eject.

### ***Current activation***

By chance, I was able to highlight the following sequence:

```
1 - change the value : hklm:\system\setup -name systemsetupInProgress = 0
2 - net stop dsmsvc ( ? in my "build," this dsmsvc service is configured with "start=demand" )
3 - net start dsmsvc
Note: 2 minutes after starting the DSMSVC service, the "Eject USB Storage" input appears if a USB
drive was connected (I use a USB boot drive for my tests)
4 - wait before changing the value systemSetupInProgress = 1
```

This Dsmsvc service is used for PDF and XPS printers. So it was while testing my printer installation that I noticed the line displaying the ejection line. It is certainly possible to separate the "Eject USB Storage" display from the printers.

### ***Current anomaly***

- Important delay: seems corrected

With the new change, the "Eject Usb Storage" line appears with a delay of about 2 minutes after the start of the DSMSVC service

- Random blockages

My BOOT USB drive is still there. I'm inserting a USB stick. It becomes visible in the "Safety Remove Hardware and Eject Media" icon at the bottom right.

From the "Eject Usb Storage" icon, if you eject the USB stick, the "explore" window is updated 1 or 2 seconds later.

But the icon at the bottom right "Safety Remove Hardware and Eject Media" doesn't show anything

until the device has been removed.

When the device is removed then the icon becomes operational again and displays my USB drive.

**But not always !**

I guess the absence of the graphic animation "Safe to remove the hardware" plays a role.

### ***Sometimes requires «start/stop DSMSVC»***

Sometimes the USB stick is visible but not the line «Eject USB Storage».

## **WPD/MTP and Smartphones**

### **A bit of history**

I've been looking for a long time to get to the point. I didn't know the WDF/WUMF architecture.

The starting point was the PNP mechanism implemented when the smartphone is connected.

The 'setupapi.dev.log' file reported an error:

```
"!!! DVI: Device not started: Device has problem: 0x25 (CM_PROB_FAILED_DRIVER_ENTRY),  
problem status: 0xc0000034."
```

After many hours of searching, I found that the driver "wudfRd" did not start.

Then long after, I found that he was looking for an ALPC port:

```
"\UMDFCommunicationPorts\ProcessManagement".
```

It still took me some time to understand that I had to look not on the side of the one who wanted to use the port but to look for the side of the one who was to create it.

And it was chance that put me on the trail of the "services.exe program".

Thanks to IDA, I found the "MiniNt" test. With Windbg I was able to prove to myself that this was the right entry point for the solution.

Then installing all the drivers and identifying all the useful files was easy.

But implementing the solution is not simple. It won't last.

- First option: change the "services.exe" program protected by "PPL security" impossible for me

Check the sites:

<http://www.alex-ionscu.com/?p=97>

<http://www.alex-ionscu.com/?p=116>

<http://www.alex-ionscu.com/?p=146>

<http://2012.ruxconbreakpoint.com/assets/Uploads/bpx/alex-breakpoint2012.pdf>

- Second option proposed by slore: use the "AppInit\_DLLs" mechanism  
<https://support.microsoft.com/en-us/help/197571/working-with-the-appinit-dlls-registry-value>

The dllentry of the hook (Nothing more):

```

case : attach to the process
    if the process is winlogon then
        delete the MiniNt key
        loop/wait : svchost process is created // services.exe test MinNt, doesn't find if,
initialize WDF, launch many services
        create again the MiniNt key
    endif
endcase

```

You can also visit the website:

<http://reboot.pro/topic/21879-is-it-possible-to-modify-the-services-exe-program-of-Winpe10-v1809/>

### ***How I installed WPD/MTP in 'Boot.Wim'***

Here I do a summary of the content of my script with the risk of error that this entails.

You have to dispose of the hook dll (see the sources below) and copy it in "system32" for example

- I use the hive software of 'Install.Wim'
- modify the Software hive to install this dll

```

Set-ItemProperty -path $cible_PS -Name "AppInit_DLLs" -Value "X:\windows\system32\hookMiniNt.dll"
Set-ItemProperty -path $cible_PS -Name "LoadAppInit_DLLs" -Value 1
Set-ItemProperty -path $cible_PS -Name "RequireSignedAppInit_DLLs" -Value 0

```

- Install multiple services/drivers and add "class"

```

Tmp_System\ControlSet001\services\WPDBusEnum
Tmp_System\ControlSet001\services\WpdUpFltr
;for SmartCard Reader (Later)
Tmp_System\ControlSet001\services\WudfPf
Tmp_System\ControlSet001\services\WUDFRd
;for MTP
Tmp_System\ControlSet001\Control\Class\{eec5ad98-8080-425f-922a-dabf3de3f69a}
;for smartCardReader
Tmp_System\ControlSet001\Control\Class\{50dd5230-ba8a-11d1-bf5d-0000f805f530}

```

- Copy files

```
$filesWPDmTP=@'
;system32
Windows\System32\WUDF*
Windows\System32\PortableDevice*
Windows\System32\Wpd*
;drivers
Windows\System32\drivers\WUDFPf.sys
Windows\System32\drivers\WUDFRd.sys
Windows\System32\drivers\WpdUpFltr.sys
;cat
Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Client-Desktop-Required-Package03111~31bf3856ad364e35~amd64~*
Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-Portable-Devices-multimedia-Package~31bf3856ad364e35~amd64~*
Windows\System32\CatRoot\{F750E6C3-38EE-11D1-85E5-00C04FC295EE}\Microsoft-Windows-WPD-UltimatePortableDeviceFeature-Feature-Package~31bf3856ad364e35~amd64~*
'@
```

- Install drivers with dism/add-driver from the DriversStore from the source

```
Dism ... /add-driver ... winusb.inf /forceunsigned
Dism ... /add-driver ... wpdmtp.inf /forceunsigned
Dism ... /add-driver ... wpdfs.inf /forceunsigned
Dism ... /add-driver ... Wpdcomp.inf /forceunsigned
Dism ... /add-driver ... wpdmtphw.inf /forceunsigned
Dism ... /add-driver ... wudfusbcciddriver.inf /forceunsigned
Dism ... /add-driver ... UsbccidDriver.inf /forceunsigned
Dism ... /add-driver ... wpdmtphw.inf /forceunsigned
copie des fichiers ".sys" dans "...drivers"
copie des fichiers "WpdMtp*.dll" dans "...System32"
```

```
copie du fichier "WpdMtpDr.dll" dans "...\System32 \Drivers\UMDF"  
copie du fichier "wudfusbccidriver.dll" dans "...\System32 \Drivers\UMDF"  
copie du fichier "wpdfs.dll" dans "...\System32"  
copie du fichier "winusb.dll 1" dans "...\System32"  
copie du fichier "winusb.sys" dans "...\inf"  
copie des fichiers ".inf" dans "...\inf"
```

## The hell of versions in Winpe illustrated with VirtualBox

It doesn't matter how useful it is to install VirtualBox in Winpe. It's an exercise like any other.

And above all it is a perfect illustration of the inconsistencies of versions of "dll" added to the system.

The VirtualBox 6,1,16-140961 version is available at the same time as Winpe 20H2.

The basis of the Winpe 20H2 files is actually the 20H1 version because MS did not release a new version of the ADK at that time.

Virtualbox requires the "OpenGL32.dll file." So I added this file to Winpe. I copied it from the ISO of Windows10 20H2.

VirtualBox software installs properly. Then, the launch of the "VirtualBox.exe program generates the error "The CheckIsMSIXPackage entry point is not found in OpenGL32.dll."

A search with "Depends.exe" shows that Winpe's "KernelBase.dll" file does not contain this entry point. The "KernelBase.dll" file of ISO 20H2 contains this entry point.

One could say this: everything is solved if we take the "KernelBase.dll" file from ISO 20H2.

But it is possible that this generates an anomaly for another component.

## Trace ETW and logman

Todo , it seems it needs the key : ...\control\wmi

utilité ?

Fichier .TMF

## Testing DotNet versions

I know little dotnet and I searched for software to test the DotNet versions present in Winpe. I chose the one from MS because I use Winpe from MS so I can use their test software well.

I found this. : <https://blogs.msdn.microsoft.com/astebner/2008/10/13/net-framework-setup-verification-tool-users-guide/>

The test tool can be extracted from a directory with the command « netfx\_setupverifier.exe /t:c:\temp /

c". Next use the test tool as well "Setupverifier2.exe /a".

It's 32-bit software but tests 64bits versions.

## Winpe mode Flat dans un VHD

## Winpe "Flat mode" in a VHD for a hyperV VM

Please visit the following sites for information on:

- Winpe's "flat mode" : <https://technet.microsoft.com/en-us/library/hh825045.aspx>
- Vhd : [https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/dd323654(v=vs.85).aspx)
- VHD and native boot :  
<https://msdn.microsoft.com/fr-fr/windows/hardware/commercialize/manufacture/desktop/boot-to-vhd--native-boot--add-a-virtual-hard-disk-to-the-boot-menu>

To use Winpe in a VHD, you need to build an adequate BCD and add the "Winpe Yes" setting. It's simple and yet ....

This site deals with the command "bootsect.exe /nt60" to start a disc:

<https://blogs.technet.microsoft.com/heyscriptingguy/2015/12/04/build-a-powershell-enabled-windows-pe-key-part-5/>

### ***Background used for the commands of this example***

For example, the paths are the following :

Boot.Wim	C:\MicroWinpeBuilder\media\sources\Boot.Wim
Fichier VHD	C:\Vhd\Winpe.vhd note : créer le répertoire si besoin
Unité pour le VHD	«H:», pour monter le VHD ( clic droit ou commande PS, ou Diskpart ou le gestionnaire de disque...)
BCD à modifier	H:\boot\bcd
Bios	Partition Bios et non UEFI ( adaptation facile ! )

### ***The five stages of construction***



**1 - Build the VHD drive** (ou vhdx) and mount it: several tools are possible

- Diskpart :

```
create vdisk file="C:\Vhd\Winpe.vhd" maximum=9000 type=fixed
attach vdisk
create partition primary
assign letter=H
format fs=ntfs quick
active
```



```
exit
```

To use a script file with diskpart :

```
copier ce texte dans un fichier « CreateVhd.txt »
lancer la commande « diskpart /s CreateVhd.txt »
```

- Powershell :

see <https://technet.microsoft.com/en-us/library/hh848503.aspx>

```
$vhdPath = "C:\Vhd\Winpe.vhd"
$vhdSize = 9GB
$vhdLetter="H"
New-VHD -Path $vhdPath -Fixed -SizeBytes $vhdSize | Mount-VHD -Passthru | Initialize-Disk -
PartitionStyle MBR -Passthru | New-Partition -DriveLetter $vhdLetter -UseMaximumSize -IsActive
| Format-Volume -FileSystem NTFS -Confirm:$false -Force
```

## 2 - Decompress 'Boot.Wim' in VHD

- 7z.exe ???

Use the graphical interface to decompress the file Boot.Wim dans H:\

- Dism

```
dism /Apply-Image /ImageFile:"C:\MicroWinpeBuiler\media\sources\Boot.Wim" /Index:1 /ApplyDir:H:\
```

## 3 - Build startup files ( legacy et UEFI) with Bootbcd.exe

```
BCDboot H:\Windows /s H: /f ALL
```

## 4 - Change the BCD with bcdedit.exe

The BCD was created by the BcdBoot command. But this order ignores Winpe. So we add this setting.

```
Bcdedit /store H:boot\bcd /set {default} Winpe YES
et avec Powershell :
Bcdedit /store H:boot\bcd /set `{default}` Winpe YES
```

We get the next BCD :

```
PS C:\WINDOWS\system32> Bcdedit /store H:boot\bcd
```

## Gestionnaire de démarrage Windows

```

-----
identificateur    {bootmgr}
device           partition=H:
description      Windows Boot Manager
locale           en-us
inherit          {globalsettings}
default          {default}
displayorder     {default}
toolsdisplayorder {memdiag}
timeout          30

```

## Chargeur de démarrage Windows

```

-----
identificateur    {default}
device           partition=H:
path             \Windows\system32\winload.exe
description      Windows PreInstallation Environment
locale           en-us
inherit          {bootloadersettings}
allowedinmemorysettings 0x15000075
osdevice         partition=H:
systemroot       \Windows
nx               OptOut
Winpe           Yes

```

 **5 – Eject VHD**

- Explorer : right click and then "eject"
- Diskpart :

```

diskpart
select vdisk file="C:\Vhd\Winpe.vhd"
detach vdisk
exit

```

- Powershell :

```
Dismount-VHD -Path "C:\Vhd\Winpe.vhd"
```

## Winpe "Flat mode" in a VHD for hard drive

As before. Only the BCD is different. In addition, it is not located in the same place.

Indeed, since we will start the PC from a hard drive (external or not, usb or not), the BCD is located in the hard drive and not in the VHD. The VHD is a simple large file contained in this hard drive. In addition, the hard drive may or may not have multiple partitions.

Example of my USB hard drive containing three F, I and J partitions, as well as several Winpe.

<b>Disque 1</b> De base 465,76 Go En ligne	<b>popReserved</b> 1000 Mo NTFS Sain (Actif, Pa	<b>winpe (I:)</b> ← 9,77 Go NTFS Sain (Partition princ	<b>Maison (J:)</b> ← 415,95 Go NTFS Sain (Partition principale)	39,06 Go Non alloué

The BCD in my USB hard drive :

```
C:\WINDOWS\system32>bcdedit /store f:\boot\bcd
```

#### Gestionnaire de démarrage Windows

```
-----
identificateur    {bootmgr}
description       Windows Boot Manager
locale            fr-fr
inherit           {globalsettings}
default           {default}
displayorder      {default}
                  {734b7449-e4bb-11e1-a114-001e330ca2a8}
                  {00a5bd49-8f29-11e6-9663-c01885b223e6}
toolsdisplayorder {memdiag}
timeout          30
```

#### Chargeur de démarrage Windows

```
-----
identificateur    {default}
device          vhd=[I:]\vhd-Winpe4-NB.vhd
path              \windows\system32\boot\winload.exe
description       vhd-Winpe4-NB
locale            fr-fr
inherit           {bootloadersettings}
osdevice       vhd=[I:]\vhd-Winpe4-NB.vhd
systemroot        \windows
detecthal         Yes
Winpe          Yes
ems               No
```

#### Chargeur de démarrage Windows

```
-----
identificateur    {734b7449-e4bb-11e1-a114-001e330ca2a8}
device          ramdisk=[I:]\sources\Boot.Wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
path              \windows\system32\boot\winload.exe
description       Winpe10 build 14393 sur partition2
locale            fr-fr
inherit           {bootloadersettings}
osdevice       ramdisk=[I:]\sources\Boot.Wim,{7619dcc8-fafe-11d9-b411-000476eba25f}
systemroot        \windows
detecthal         Yes
Winpe          Yes
```

ems	No
Chargeur de démarrage Windows	
-----	
identificateur	{00a5bd49-8f29-11e6-9663-c01885b223e6}
<b>device</b>	<b>vhd=[J:]\mesvhd\vhd2016\test.vhd</b>
path	\windows\system32\boot\winload.exe
description	MicroWinpeBuilder 14393
locale	fr-fr
inherit	{bootloadersettings}
<b>osdevice</b>	<b>vhd=[J:]\mesvhd\vhd2016\test.vhd</b>
systemroot	\windows
detecthal	Yes
<b>Winpe</b>	<b>Yes</b>
debug	Yes
ems	No



Attention about ramdisk:

Just because you'll write « **ramdisk=[I:]\sources\Boot.Wim,{7619dcc8-fafe-11d9-b411-000476eba25f}** » does not mean that the "...boot-bcd" file contains information about the registry entry « {7619dcc8-fafe-11d9-b411-000476eba25f} ». Make sure you're either where the "BCD" file came from or the presence of that entry.

This MS site explains this point as well as how to verify this entry.


<https://social.technet.microsoft.com/Forums/en-US/f2f2cabe-27b4-4cea-bf1d-76d4cea7ccbf/alphnumeric-object-in-the-dvd-bcd?forum=win10itprosetup>


<b>bcdedit /store "C:\Users\Balubeto\Downloads\DVD\boot\bcd" /v /enum all</b>	
...	
Device options	
-----	
identifiant	<b>{7619dcc8-fafe-11d9-b411-000476eba25f}</b>
ramdisksdidevice	boot
ramdisksdipath	\boot\boot.sdi

This entry is not present in the BCD located on your PC's hard drive.

## FbWf et PXE

## The RamDisk FBWF. SYS: ScratchSpace limited to 512MB?

 I'm not really interested in this feature. The signature mechanism introduces the difficulty to circumvent. That's what I think is useful to test

 projects like WIMBUILDER2/XPE/SE use an old "ramdisk" that comes from an old MS product that has become obsolete ("Windows Embedded"). But his 'fbwf.sys' file continues to work in Winpe of W10.

Can DISM be used to change this maximum size of 512MB? If we try:

```
«dism /image:%Mount% /Set-ScratchSpace:1024»
```

one obtains: "Valid workspace value. Select 32, 64, 128, 256 or 512."

I found no other way but to change the driver's code. After the code has been changed, the "checksum PE" data must be changed, the file must be signed and the BCD changed.

To disassemble the driver's file, I used "PeBrows64 Pro". Check the site for more details :

<http://www.smidgeonsoft.prohosting.com/>

I now use IDA V7 (now free) to disassemble the driver's file.

### ***A little reading with IDA***

[I'm making corrections with the 20h2 version because there have been changes]

The entry point is « driverEntry ». We quickly find the function « FbwfInstanceSetup » with the text « FbwfInstanceSetup: Failed to adjust scratch space; status = 0x%X ». I'm going to dig a little. A call to « ZwQuerySystemInformation », a calculation, a new comparison with « CMP RAX,0x40000000 » followed by a "JBE" to the error text ...

I change the "JBE" to JMP to neutralize the jump to the error code...

And it works, the size of the cache in writing for X: is 2GB in the Winpe with at least 4GB of RAM.

### ***Change the "checksum PE"***

64bit driver files must display a valid checksum.

### ***The signature for testing***

You have to install part of the SDK to find the 3 'magic' programs: Makecert.exe, CertMgr.exe, SignTool.exe. It's a little long.

With windows10 64bits, each driver must be signed either with a 'real' certificate or with a test

certificate.

A driver developer has 2 possibilities for his (repetitive) tests:

- block the signature check on the test PC

MS prevents the installation of such an unsigned driver anywhere and by anyone. The developer must act physically on the test PC to block the verification of a driver's certificate with:

- support F8 when starting and selecting the option ...
- either by connecting a debugging PC ... the driver loading requires the developer to press 'g' into his PC
- use a test certificate deployed on multiple test PCs

This will also require changes to the BCD with the "TESTSIGNING" option. 'Mode test' information will appear at the bottom right on the screen. Thus the end user is warned of the danger.

Example : Bcdedit.exe /store ??:\boot\bcd -set {default} TESTSIGNING ON

For more information on « The TESTSIGNING Boot Configuration Option » :

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff553484%28v=vs.85%29.aspx?f=255&MSPPErr=-2147217396>

Note :

Before setting BCDEdit options you might need to disable or suspend BitLocker and Secure Boot on the computer.

Starting with Windows 7, Windows displays this watermark only in the lower left-hand corner of the desktop.

### ***To sign the modified driver***

You have to create a test certificate, eventually install it on the developer's machine, and sign the file.

### ***Creating a test certificate with 'Makecert.exe'***

For more information:

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff548693\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff548693(v=vs.85).aspx)

MakeCert -r -pe -ss TestCertStoreName -n "CN=CertName" CertFileName.cer

My current order ( 20h2 ) :

```
Cd « c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64 »
MakeCert.exe -r -pe -ss NoelBlancStore -n "CN=CertificatTestingNoelBlanc" CertificatTestingNoelBlanc.cer
```

## ***Installing the certificate in the developer's pc***

For more information:

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553506\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553506(v=vs.85).aspx)

It should be remembered that:

- to sign a driver: the certificate can be installed in any store
- but to verify the signature of a driver signed with this certificate, this certificate must be installed in: "Trusted Root Certification Authorities"

The name of the 'Trusted Root Certification Authorities certificate' store is root.

Before signing a file, you must install the certificate on the developer's machine. If the certificate was generated on the same machine, is it useful to launch the next comande?

Hence my commands, being admin:

```
Cd « c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64 »  
CertMgr.exe /add CertificatTestingNoelBlanc.cer /s /r localMachine root
```

The certificate is visible with 'Certmgr.msc' in my pc of 'dev'

## ***But how do you sign the pilot with this certificate?***

Test-Signing a Driver File :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553467\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553467(v=vs.85).aspx)

Test-Signing Driver Packages :

<https://msdn.microsoft.com/en-us/library/windows/hardware/ff553480%28v=vs.85%29.aspx?f=255&MSPPErr=-2147217396>

The following command shows how to use SignTool to test-sign a driver file.

This example embeds a signature in Toaster.sys, which is in the amd64 subdirectory under the directory in which the command is run.

The test certificate is named "contoso.com(test)" and it is installed in the certificate store named "PrivateCertStore."

```
SignTool sign /v /s PrivateCertStore /n contoso.com(test) /t  
http://timestamp.verisign.com/scripts/timestamp.dll amd64\toaster.sys
```

Avec 20H2 :

<https://docs.microsoft.com/fr-fr/windows-hardware/drivers/install/test-signing-a-driver-file>

Email : noelblanc.Winpe@free.fr



new one : /t http://timestamp.digicert.com

Hence my command:

```
Cd « c:\Program Files (x86)\Windows Kits\10\bin\10.0.17763.0\x64 »
SignTool sign /v /s NoelBlancStore /n CertificatTestingNoelBlanc /t http://timestamp.digicert.com
fbwf.sys.new
```

### ***Installing the certificate on the test machine (not Winpe)***

This procedure is to be used in the case of a non-Winpe windows.

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff547618\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff547618(v=vs.85).aspx)

The test certificates that are used to embed signatures in driver files and to sign a driver package's catalog file must be added to :

- the Trusted Root Certification Authorities certificate store
- the Trusted Publishers certificate store.

Installing a Test Certificate on a Test Computer :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff553563\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff553563(v=vs.85).aspx)

The following CertMgr command adds the certificate in the certificate file CertificateFileName.cer to the Trusted Root Certification Authorities certificate store on the test computer:

```
CertMgr.exe /add CertificateFileName.cer /s /r localMachine root
```

The following CertMgr command adds the certificate in the certificate file CertificateFileName.cer to the Trusted Publishers certificate store on the test computer:

```
CertMgr.exe /add CertificateFileName.cer /s /r localMachine trustedpublisher
```

### ***Installing the modified and newly signed driver in Winpe***

There is no need to install the certificate in Winpe.

We're copying the 'FbWf.sys' file in Winpe.

And don't forget to change the desired value for the size of the RamDisk:

```
HKEY_LOCAL_MACHINE\SYSTEM\ControlSet001\services\FBWF
```

```
WinPECacheThreshol=dword 00000400 si 1Go
```

```
WinPECacheThreshol=dword 00000800 si 2Go
```

### ***Change the BCD to add « TESTSIGNING***

Then the BCD is changed to activate the option "TESTSIGNING".

Not needing FBWF in a VHD, I copy the original file when I build the VHD.

### ***Special feature for the UEFI environment***

I use VM in HyperV. To start Winpe with a modified driver and signed with a test certificate, you have to disable "Secure Boot" in the VM settings. I did not test with a real PC and UEFI but I guess you also have to disable this option.

Note: The "NoIntegrityChecks" BCD setting has no effect on the boot.

### ***Some sites***

Vhd : <http://go.microsoft.com/fwlink/?LinkId=205691>

Bcdedit :

<https://msdn.microsoft.com/en-us/windows/hardware/commercialize/manufacture/desktop/bcdedit-command-line-options>

[https://technet.microsoft.com/en-us/library/cc709667\(v=ws.10\).aspx](https://technet.microsoft.com/en-us/library/cc709667(v=ws.10).aspx)

[https://msdn.microsoft.com/en-us/library/windows/hardware/dn653287\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/dn653287(v=vs.85).aspx)

## **Winpe's pxe boot with tftp32 64bits version of "Jounin"**

This is too slow with a 'Boot.Wim' image of more than 1GB.

The following information is there to help my memory.

Download the TFTP server from 'Jounin'.

<http://tftpd32.jounin.net/download/tftpd64.452.zip>

The DHCP of an MS server could also be used.

Files must be retrieved from the 'Boot.Wim' image. So put together this image. And copy the files <mount>\windows\system32\boot\pxe in a directory that will be the root of the TFTP server. Then copy the contents of the USB key, so the whole tree files of your final Winpe (boot, sources, etc.) in this root of the TFTP server.

The configuration of the TFTP server is quite simple: check TFTP server and DHCP server, inform the file to load "pxeboot.n12", put a fixed ip possibly on the pc hosting the tftp server, inform the basic directory (the one that hosts all the previous files), open the log tab.

Think about opening "public domain" in the firewall for TFTP, which should be done because the first launch of TFTP64 of "jounin" opens these ports (UAC?), and the firewall should have warned you (depending on your configuration of course).

Basically: connect the two PCs directly via a cable "rj45" (no need for cross-wire with the gigabyte standard of all modern pc), modify the "bios" to activate the "pxe boot". When starting the pc, we choose "boot pxe." After requesting an ip address from the DHCP server, the card sends a TFTP frame to the network. The TFTP server on the other pc receives it and returns the file "pxeboot.n12" (n12 or com depending on whether or not you want to act on F12 along the way but I do not remember what it is for). Then the pc runs this program. The latter then loads 'bootmgr.exe' from the TFTP pc server.

Email : [noelblanc.Winpe@free.fr](mailto:noelblanc.Winpe@free.fr)

This is "bootmgr.exe" and not "bootmrg" from the root of the USB key! Then bootmgr.exe" in turn loads the BCD, then everything else ('boot.sdi, 'sources'Boot.Wim') from the TFTP server. It's a long time, but it works.

## Investigation part within Winpe

## Basic tools: procmon64.exe, procex64.exe, depends.exe, SPY, Windbg, graphedit

With Winpe in 64 bits, you have to use the 64-bit versions of the software (unless the 32-bit subsystem has been installed).

The 64-bit version of the "sysinternal" tools is sometimes integrated (depending on the versions) in the file downloaded from Technet but is invisible at first. It is revealed by the 32-bit version that makes it visible during its use.

- Understand Procmon's contextual menus:
  - "Stack" : provides information about the dll (and thread) that triggered the current action. This can guide the search.
  - Colonne TID = « Thread » ID, to better understand the contents of the battery
  - Don't forget the symbols
  - "Property" : allows you to know the command line, which is useful and rich in information.
- The configuration file: it increases comfort

### ***A direct observation with Procmon***

An example to fix the ideas: add MMC.exe, the empty shell that launches the snappins.

- We launch Procmon and we launch mmc.exe.
- Quickly a message appears indicating an error.
- We're stopping the capture.
- We examine the trace of Procmon: incomprehensible!

But you learn fast.

A classic mistake: too many filters. What is missing is sometimes, but not always, in another process called by the target process. So be careful with the filters.

Generally, you have to go towards the end. But since the display also includes information in the track, you have to learn to identify it in order to get the right information. There, it's experience and knowledge!

And from near to near, from correction to correction (added a dll or program identified as missing, a COM object key, etc.) we manage to identify all the missing files/keys.

When 'MMC.exe' start without error, we have a list of everything to add, keys, files (including assembly).

☛ The limits of investigation with Procmon: calls to services, drivers, COM objects, "named pipes", etc., are not traced.

☛ Other almost invisible objects: the objects in the system are visible with 'WinObj.exe' of 'technet-sysinternals'. See my script that visualizes these objects.

### ***Symbols with Procmon***

- dbghelp.dll and symsrv.dll

The best solution is to download the debugger (ADK or DDK), place its repertoire in the Winpe root, and modify Procmon's options to point to this directory

A less good solution is to place the two dll, 'dbghelp.dll' and 'symsrv.dll', in the same repertoire as protmon.exe. Both dll are from the ADK/debugger.

So Procmon will pick up the symbols in the MS server.

And with an operational network, the consultation of the "stack" becomes more explicit and gives better information.

- local cache

To have a local cache ( here [g:\symcache](#) ), you have to change a Procmon option, menu « Configure Symbols/symbols Path » like this( without space ) :

```
srv*g:\symcache*http://msdl.microsoft.com/download/symbols
```

The symbols will be downloaded from the remote site and stored in the cache directory on the disk to speed up future displays.

See : <https://developer.microsoft.com/en-us/windows/hardware/windows-driver-kit>

- Session System

Procmon displays the message "loading symbol ... but the download does not take place and therefore no display.

The local cache can be useful. We start by doing the same consultations with the ADM session and the cache fills up. Then we start again with the System session: procmon consults the previously filled cache.

After much research, I finally found the right explanation and solution. You have to create the key:

```
reg add « hklm\software\Microsoft\Symbol Server » -v NoInternetProxy -v 1 -t Dword
```

## Capture from start-up

<https://blogs.msdn.microsoft.com/scw/2015/10/27/how-to-enable-boot-logging-in-windows-10/>

Introducing the mechanism used:

The 'Procmon24.Sys' driver is installed with the 'promon.exe' menu ( options/Enable Boot Logging).

When restarting the microcomputer, this driver is active very early and can capture all the events of the start-up.

It can be stopped and events saved:

- interactively with Procmon's classic graphical interface,
- to check : or with the following command line to place in a script for example:

```
C:\procmon\Procmon /BackingFile C:\procmon\log.pml /AcceptEula /Quiet /noconnect
```

☛ During capture, the driver filled out a temporary file in the directory « ... \windows\procmon.PMB »

At the end of the capture, during the backup, the driver is deleted. Sometimes he stays there. And if it stays active on the next boot in the VHD, it will produce a new file that could fill that VHD.

With Winpe, you have to

- install this driver by creating the entrances to the Hive System,
- copy the 'Procmon23.sys' driver in the directory « ... \windows\system32\drivers ».

To dispose of the driver, you have to start a capture with Promon and copy the file into a backup

directory.

To have the necessary inputs to create in the Winpe hive, it is easier to implement this capture and make an "export" with regedit.

In the event that Winpe is used in Flat mode with a VHD virtual disk, and for successive tests, the driver should be checked and copied again in ...windows\system32\drivers.



Tip implemented to stop the capture this procmon: a cmd is launched with Winpeshl. This CO launches the "timeout.exe" program. At the end of this period, the stop-capture order is launched.

## **BSOD: the advantage of VHD**

Le VHD permet de générer un fichier de Dump que l'on peut ensuite analyser avec Windbg.

If my observations are correct, the dump file is created on the next boot. The Bone restarts and exploits the "pagefile.sys" file that contains the details of the error. And if you don't want or can't reboot, it's possible to take the "pagefile.sys" file and analyze it with WinDbg.

## **Windbg: too much information to write for now**

symboles

Kernel debugger network

Kernel debugger hyperV

## **Investigative and construction strategies**



### ***The logic of the "ant": unitary elements***

For each feature, we look for useful items one by one. And we build large lists of keys and files for each feature.

For greater efficiency, it is possible to capture the changes made by installing a component. But that's not always possible.

This search for detail quickly becomes tedious.

Moreover, when many components have been added, the size of the software hive thus completed is almost equal to that of the reference.

### ***The logic of the "elephant": global elements***

Starting from the principle of "who can most can least", and having found for example that the keys of the ClassRoot hive are very useful, one can copy whole sections of register without asking too many questions, as long as one understands a little what one is doing.

### ***Pieces from the "Software" hive.***

From a well-chosen source (a downloaded and mounted iso, a PC under Win10 active) we export pieces of hive. For example, CLSID, APPID... or even ClassRoot in its entirety.

The list becomes shorter but we lose a little knowledge of the detail.

The first pitfall is the loss of data during export because some keys are protected by ACL.

- WinPeSe method: first and foremost, modify the ACL of the hives.
- Identify error-generating ACL with Procmon

It is also possible to export from Winpe by loading the file of the hive to be analyzed. In this Winpe environment, ACL does not cause any problems, does not generate errors.

We must not forget then to replace inappropriate values such as paths, for example, « C:\ » to be replaced by « X:\ ».

Watch out for the hive "currentuser".

### ***The entire "Software" hive: the only case handled by my scripts!***

You can also take the whole "Software" hive in the "Install.Wim" file. We must not forget to:

- remove the "runas" in the keys «...\classes\appid»
- copy the key ««...windows Nt\Winpe» dfrom the WinPe hive in the destination hive from «Install.Wim».

Also, don't forget that this hive also contains the keys to the 32-bit subsystem.

☛ A pitfall already encountered: I was sometimes forced to put the "sideByside" key of Winpe back in the hive "software" from 'Install.Wim' because Winpe versions of the ADK are not always updated with the ISO of MS.

### ***The "System" hive***


One may also wonder if we can take the entire System hive in Install.Wim.

I cannot find the link that explains these changes.

Mais je sais qu'il faudra faire quelques modifications : modifier System\Setup\SetupPhase, Cmdline and disable some drivers.

The value of SetupPhase corresponds to the phases described in the ADK documentation, Audit, Specialize... It guides Winpe's behavior, winlogon software in particular.

Don't forget the role of the ProductPolicy key.

 System\control...\control\MiniNt and «SystemSetupInProgress» keys are an indicator, of the activity of a Winpe Os. They are tested, for example, by eventvwr.msc, « remote powerhell», spooler, termsrv, dsmsrv...

### ***The "Drivers" hive***

To facilitate the addition of drivers and remove all obstacles, I ended up using the hive "Drivers" of 'Install.Wim' in full.

To check: what would be the advantage of copying the key as well:

HKEY\_LOCAL\_MACHINE\SYSTEM\DriverDatabase

### ***The files***

We could copy everything there too, but it will be a very, very large volume. And don't forget the ".mui" files (related to the available languages). The "assemblies" of "Dot Net", the drivers, the "driverStore", would thus be available. In a Vm, this can be envisaged.

But too many files are useless.

### ***Comparisons***

For simple features, it's sometimes useful to compare tracks taken on an active Windows10 and Winpe. For example, on an active Windows10, with the task manager, you end exploring. We activate "Procmon." And we relaunch "Explore.exe" from the task manager. We do the same with Winpe and compare. This can help sometimes.

You have to find the test, the comparison that will give relevant information.

## **Migrating to a new version of ADK?**

When a new version of WinPe is available, the Winpe builds on the basics of the old script versions of MicroWinpeBuilder has little chance of working the first time. At the start, you can sometimes see that the screen turns black and everything seems frozen! Indeed, new files/keys appear and others disappear in the supply of MS. Hence the need for an effective method to identify differences. Of course Procmon remains an effective tool but faced with a black screen, how to interact with this program that does not work, how to discover it?

### ***BSOD***

It is best to use a VHD to benefit from the analysis of the dump file.

### ***An all-black screen: the worst.***

Remember that the Themes feature:

- displays the title bars of the windows with a new aesthetic (square corner ...)

- uses CoreMessagingRegistrar and Themes services

When starting, 'Winlogon.exe' implements the DWM mechanism and starts it with 'dwminit.dll' if that file is present.

By activating the capture at the start of Procmon, we can identify the components that are useful for displaying the desktop.

If all components useful to DWM are not identified and available, the screen will remain black.

Don't forget to check the presence of "D2D1.dll.mui" in the language directory: if not black screen.

You can do it in stages, neutralize 'dwminit.dll' to disable DWM and check the services, and then hand over dwminit.dll.

### ***Anomaly when opening the ADM session***

We can verify that the "console mode" mechanism works. To do this rename Windows.ui.logon.dll.

## Part MicroWinpeBuilder Scripts

# Construction scripts

## **Preamble**

- The language of The WinPe builds: it will be the language of the ISO image.
- Adk, ISO and host : identical version. I have not tested in another configuration.

## ***The 'Boot.Wim' file used is generated by the ADK of windows10***

Current scripts use the ADK tree for WinPe. The "Media" directory is therefore the "Boot.Wim" file.

The script only once starts building the "Boot.Wim" file using "cotype.cmd."

This file contains all the packages offered by MS. Once built by the ADK, it is backed up with the name "Boot.Wim.WithPaquetsDeBase.export". It is this last file that will be copied into "Boot.Wim" in order to be modified by the scripts.



It is possible to take a file's 'Boot.Wim' already created by another tool (for example by WinBuilder) and edit it with new scripts (no interest in taking the current scripts). To do this, it must be renamed as a Boot.Wim.WithPaquetsDeBase.export.

## ***The host machine***

You have to use a machine on Windows10. Various programs, such as DISM when adding drivers, do checks on .cat files that may not work with another OS.

## ***The two reference sources: ADK and Win10***

### ***ADK***

You have to download and install the ADK for windows 10. Scripts support the launch of cotype.cmd.

### ***ISO of windows10***

The ISO image of windows10 contains an "Install.Wim" file.

This "Install.Wim" file must be decompressed in a reference directory.

It's up to you to edit the scripts if you want to automate this task.

## ***Summary description***

### ***GUI***

A first script offers a rudimentary GUI to validate the three necessary paths:

- the directory that "cotype.cmd" will be used. It must not exist before the launch of cotype.
- the basic directory containing the ADK.

- the directory containing the decompressed reference "Install.Wim."

This GUI proposes the recording of this data in a file "attached" to the script (alternative stream).

Treatments will be visible in the "console" tab. A log file will be produced at the end of processing.

Not all errors are dealt with!

### ***Traitement.PS1***

This is the most complex main part. It takes the following steps:

- launch of "copyype.cmd" to build the base repertoire tree
- launching "dism" to add all the packages of the ADK: it's very long!
- adding all the identified changes.

The first two steps are completed only once as long as the "Boot.Wim.WithPaquetsDeBase.export" file exists. Subsequent script launches use a copy of this file. They only do the third step.

To start over from the beginning: delete WinPe's core directory.

### ***The basic tree used by scripts***

...\MicroWinpebuilder

This is the directory in which the ADK comes to deposit its files  
it contains log and iso

...\MicroWinpebuilder\Add\_Drivers

it contains drivers for my USB printer

...\MicroWinpebuilder\Add\_Root

it contains procmon, winobj, etc

...\MicroWinpebuilder\MesProfilsWifi

it contains the files used for wifi (different geographical locations)

...\MicroWinpebuilder\Media

it is generated by the ADK

Scripts are designed to edit 'Boot.Wim'

...\MicroWinpebuilder\ISO

copy of ...media to build ISO file

### ***Added to the tree***

- Drivers :

2 solutions, change the script code 'processing.ps1' or 'Dism add-driver' after setting up the 'Boot.Wim'

Email : noelblanc.Winpe@free.fr

file

- Script in the root :

either by modifying the "ihm" and "processing" scripts if the file is present in the same directory as the "GUI" script, or by placing it in the directory « ...\MicroWinpebuilder\Add\_Root »

- Printers

See the script because each case is unique

### ***The structure of a component to be added***

For any component, the same elements are always found:

- files
- registry keys
- Acl
- Delayed treatments when starting Winpe (to make it simple and quick)

To optimize the registry loads/unloads and avoid a conflict with "DISM", I was led to burst processing a component. The logic of the script becomes:

- Preparing post-processing files
- Loading the hives
- For each component, changes to the registers
- Unloading the hives
- For each component, copies of files
- Dism for the addition of driver

I recognize that modifying or adding a component quickly becomes complicated under these conditions. But with a little time...

### ***Additional files : ProductOptions, English, ... TODO !***

#### **ProductOptions**

It contains data about programs authorized in Windows 10.

#### **English**

It is the translation of the manual.

### **Script de connexion WifiConnexion**

I took over a program found here : <http://managedwifi.codeplex.com> . I deposited it in a PS script with a GUI. It is functionally identical to the "Netsh" controls.

Email : noelblanc.Winpe@free.fr

I added the possibility of disconnection.

☛ the refresh with these APIs is quite long, sometimes 30 seconds.

## Script IHM\_Get-Set\_WinpeScreenResolution.ps1

A copy/paste of code taken from the internet. At the end of the change, sending the message «WM\_SETTINGCHANGE».

## Get-checksumPrg

It was useful for me to edit 'fbWf.sys'

It allows you to :

- control a program's checksum information by comparing
  - the checksum inscribed in the file by the linker
  - and the checksum calculated by an OS API
- modify this checksum if need be ( param -modify true )

There are several APIs to control these two checksum. I take the one I've already used:

MapFileAndChecksum de imagehlp.dll

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms680355\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680355(v=vs.85).aspx)

The checksum modification will be written in the program file.

The description of the PE format of a file remains unavoidable.

Sites :

[https://fr.wikipedia.org/wiki/Portable\\_Executable](https://fr.wikipedia.org/wiki/Portable_Executable)

[https://msdn.microsoft.com/en-us/library/windows/desktop/ms680336\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/desktop/ms680336(v=vs.85).aspx)

## SearchSignature

I had quickly written a piece of script in PS. The files to be modified being small (less than 500KB), the treatment with Powershell was suitable. Over the years, I've taken the time to include a little C-code in the PS script to speed up processing.

And as now I have chosen to modify several programs, it is integrated into my big script.

But I keep a standalone version to do some spot checks.

## The hook dll for WPD/MTP

```
// dllmain.cpp : Defines the entry point for the DLL application.
```

```
#include "stdafx.h"
```

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

```
#include <TIHelp32.h>
```

```
HANDLE OpenLog();
```

```
void WriteLog(HANDLE hFile, TCHAR *text);
```

```
LSTATUS SimpleDelay();
```

```
// le main
```



```

BOOL APIENTRY DllMain(HMODULE hModule,
    DWORD ul_reason_for_call,
    LPVOID lpReserved)
{
    switch (ul_reason_for_call) {
    case DLL_PROCESS_ATTACH:
        SimpleDelai();
        break;
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}

DWORD FindAppProcessID(HANDLE hFile, TCHAR *strAppName)
{
    TCHAR buff[512] = { 0 };
    HANDLE handle = CreateToolhelp32Snapshot(TH32CS_SNAPALL, 0);
    PROCESSENTRY32 Info;
    Info.dwSize = sizeof(PROCESSENTRY32);
    if (Process32First(handle, &Info)) {
        do {
            TCHAR *ss = Info.szExeFile;
            wsprintf(buff, TEXT("%d:%s"), Info.th32ProcessID, ss);
            WriteLog(hFile, buff);
            if (wcscmp(ss, strAppName) == 0) {
                CloseHandle(handle);
                return Info.th32ProcessID;
            }
        } while (Process32Next(handle, &Info));
        CloseHandle(handle);
    }
    return 0;
}

// "winlogon" is loaded before "services".
// for winlogon only (in dllmain, reason == DLL_PROCESS_ATTACH):
// We kill the Minint key. We wait for svchost.exe process.
// after delay, we create the minint key

LSTATUS SimpleDelai()
{

```

```

TCHAR szFileName[MAX_PATH] = { 0 };
GetModuleFileName(NULL, szFileName, MAX_PATH);
HKEY dmy;
if (wcsncmp(szFileName, L"X:\\windows\\system32\\winlogon.exe", 32) == 0) {
    RegDeleteKey(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\
Control\\MiniNT");
    TCHAR buff[512] = { 0 };
    HANDLE f = OpenLog();
    SYSTEMTIME sys;
    GetLocalTime(&sys);
    wsprintf(buff, TEXT("%4d-%02d-%02d %02d:%02d:%02d.%03d"), sys.wYear,
sys.wMonth, sys.wDay, sys.wHour, sys.wMinute, sys.wSecond, sys.wMilliseconds);
    WriteLog(f, buff);
    for (int i = 0; i < 30; i++) {
        GetLocalTime(&sys);
        wsprintf(buff, TEXT("%4d-%02d-%02d %02d:%02d:%02d.%03d"), sys.wYear,
sys.wMonth, sys.wDay, sys.wHour, sys.wMinute, sys.wSecond, sys.wMilliseconds);
        WriteLog(f, buff);
        if (FindAppProcessID(f, TEXT("svchost.exe")) > 0) break;
        Sleep(500);
        RegCreateKey(HKEY_LOCAL_MACHINE, L"SYSTEM\\CurrentControlSet\\
Control\\MiniNT", &dmy);
        CloseHandle(f);
    }
    return ERROR_SUCCESS;
}
HANDLE OpenLog() {
    HANDLE hFile = 0l;
    TCHAR szFileName[MAX_PATH] = { 0 };
    hFile = CreateFile(L"X:\\hook.dat", FILE_APPEND_DATA, FILE_SHARE_READ |
FILE_SHARE_WRITE,
        NULL, OPEN_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
    return hFile;
}
void WriteLog(HANDLE hFile, TCHAR *text)
{
    SetFilePointer(hFile, 0l, NULL, FILE_END);
    DWORD dwWritten;
    TCHAR buff[512] = { 0 };
    wsprintf(buff, TEXT("%s\r\n"), text);
    if (hFile) {
        WriteFile(hFile, buff, lstrlenW(buff) * sizeof(TCHAR), &dwWritten, NULL);
        FlushFileBuffers(hFile);
    }
}

```

}  
}

# ARCHIVES

# ARCHIVES

## DCOM et la bascule vers l'interface graphique classique

### Ne fonctionne plus depuis V1703

La nouvelle interface graphique appelée «Metro» semble impossible à mettre en œuvre dans Winpe. Néanmoins certaines actions sont définies par la valeur d'un pointeur dans le registre. Il est parfois possible de modifier ce pointeur et d'activer l'interface classique d'un composant.

Une comparaison avec windows7 permet d'identifier les divergences relatives à la nouvelle interface.

Par exemple, le clic droit sur le bureau permet de sélectionner les éléments «Paramètres d'affichage» ou «Personnaliser» du panneau de configuration. Ceci déclenche l'affichage de la nouvelle interface. Une modification judicieuse permet de retrouver l'ancienne interface.

- Nouvelle interface 'metro' :

```
HKCR\DesktopBackground\Shell\Display\command\DelegateExecute={556FF0D6-A1EE-49E5-9FA4-90AE116AD744}
```

qui active l'objet COM : CLSID\_LaunchSettingsPageHandler

- Retour à l'ancienne interface :

```
HKCR\DesktopBackground\Shell\Display\command\DelegateExecute={06622D85-6856-4460-8DE1-A81921B41C4B}
```

qui active l'objet COM :COpenControlPanel

Mais dans le cas de l'affichage, on est confronté au comportement décrit au chapitre suivant.

## DCOM sur 64 bits : un bug avec Explorer ?

Dans Winpe 64bits, si on lance «Desk.cpl» par exemple, on constate dans le gestionnaire de tâches que le logiciel dllhost.exe est chargé. Ce programme est un «surrogate» faisant partie de l'architecture DCOM. Aucun affichage n'a lieu. Un nouveau lancement de «Desk.cpl» charge un nouveau processus «dllhost.exe». Le dialogue client-serveur DCOM semble impossible.

Dans les mêmes conditions avec un Windows10 64bits, avec «Procmon», on constate le chargement de «x:\windows\syswow64\dllhost.dll» puis rapidement son déchargement. Il s'agit bien de la version 32 bits de 'dllhost'.

Or, dans la version classique de Winpe64 de l'ADK, le sous-système 32 bits est absent. D'où l'anomalie.

Est il possible d'imposer à «Explorer» de charger un serveur DCOM 64 bits de cette fonctionnalité ?

Sites à consulter :

Securité dans COM : [https://msdn.microsoft.com/en-us/library/ms693319\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms693319(v=vs.85).aspx)

dllhost : [https://msdn.microsoft.com/en-us/library/ms695225\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms695225(v=vs.85).aspx)

APPID : [https://msdn.microsoft.com/en-us/library/ms678477\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/ms678477(v=vs.85).aspx)

<https://blogs.msdn.microsoft.com/jigarme/2007/10/09/what-is-appid/>

Com-32-64 : <http://mariusbancila.ro/blog/2010/11/04/32-bit-and-64-bit-com-servers/>

## Wow64 : le sous-système 32 bits dans Winpe

Un site chinois (mais j'ai oublié lequel) explique que Winlogon crée normalement 2 objets system pour le fonctionnement du sous-système 32 bits.

Si j'ai bien compris l'histoire de cette fonctionnalité, ces deux objets n'étaient plus créés avec certaines versions de Winpe.

Des développeurs chinois avaient donc écrit les logiciels « SetWow64.exe » et « LoadWoW64.exe ». Ces logiciels sont repris par l'équipe de WinPeSe.

Or, depuis peu (Version 1709), Winpe assure la création de ces deux objets System. Donc ces logiciels ne sont plus utiles.

### **Que sont les objets system ?**

Chercher sur MSDN et consulter les sites ci-dessous.

En français :

<http://www.ivanlef0u.tuxfamily.org/?p=39>

Object Directories :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff557755\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff557755(v=vs.85).aspx)

NtCreateDirectoryObject :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff556456\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff556456(v=vs.85).aspx)

ZwCreateDirectoryObject :

[https://msdn.microsoft.com/en-us/library/windows/hardware/ff566421\(v=vs.85\).aspx](https://msdn.microsoft.com/en-us/library/windows/hardware/ff566421(v=vs.85).aspx)

### **Pour explorer les objets system : winobj.exe et Procexp.exe**

Il faut le télécharger depuis le site 'technet/sysinternals'. Il existe en version 32bits uniquement.

Il me semble qu'il ne donne pas de bons résultats en 64 bits.

Procexp avec «Ctrl+H» me semble plus efficace. Mais il ne donne pas un aperçu global.

## Script MonSetWow64 pour le sous-système 32 bits

**Devenu inutile au moins depuis V1709**

## **L'origine**

L'équipe WinPeSe propose un programme qui permet de mettre en œuvre le sous-système 32 bits. Ne voulant pas utiliser de programme externe, j'ai repris les sources écrites par un développeur chinois et je les ai converties en C#. Puis je les ai déposées dans le script pour les rendre lisibles.

## **Le principe**

Lu sur internet :

yamingw found a solution for WoW64 in Win10PE

<http://bbs.wuyou.net/forum.php?mod=viewthread&tid=371490>

<http://winbuilder.cn/forum.php?mod=viewthread&tid=204>

Run setwow64. Ntoskrnl is phase 1 initialization testing if the WinPE is running in memory, it does not create a KnownDlls32 kernel objects. This object is populated with SMSS. When initializing a 32-bit system this object was not found in the path is wrong. This procedure only creates a path, no fill. Source refer to the <https://github.com/Google/SymbolicLink-testing-tools>, the CreateObjectDirectory and NativeSymLink merged. Kernel objects when the application exits disappear, so will both fill in Winmain.

SetWOW64 by yamingw. It works by creating the KnownDlls32 kernel object and linking it against X:\Windows\SysWow64 folder. It ~does what smss.exe should do. Note that it does not allow ThinApp packages to work.

## **Visualisation**

Depuis un environnement 64 bits, on peut lancer ce script aussi bien dans WinPe que dans Windows10. Sans paramètre, il permet une visualisation des « objets system ». Il est rudimentaire et demande à être amélioré. Mais tel quel, lancé dans WinPe 64bits, il permet d'identifier d'éventuels objets manquants.

Note : winobj.exe de systinternals n'existe pas en version 64bits. Un autre programme du même nom existe mais je ne souviens pas du site où il est disponible.

## **Création**

Avec le paramètre « create », il crée les objets nécessaires pour faire fonctionner le sous-système 32 bits.

## **Modification du wallpaper et de themecpl.dll**

**Ne fonctionne plus depuis V1703**

## **Le contexte**

La première étape a consisté à rendre possible l'accès au menu «Personnaliser» ("Personalization"). Merci à ceux qui ont trouvé le mécanisme de changement de GUID dans la clé "...\DesktopBackground".

Ce menu est accessible à partir d'un clic droit sur le bureau .

On peut ouvrir la fenêtre «Personnaliser» ("Personalization") avec la commande :

```
control.exe /name Microsoft.Personalization
```

La fenêtre «Personnaliser» ("Personalization") offre en bas un lien "Desktop Background" qui permet de changer le wallpaper.

On peut ouvrir cette fenêtre avec la commande :

```
control.exe /name Microsoft.Personalization /page pageWallpaper
```

On peut aussi ouvrir la fenêtre «Couleur» ("Color") avec :

```
control.exe /name Microsoft.Personalization /page pageColorization
```

Sites :

une porte d'entrée mais il faut remonter dans l'arborescence du site :

[https://msdn.microsoft.com/en-us/library/bb773213\(VS.85\).aspx](https://msdn.microsoft.com/en-us/library/bb773213(VS.85).aspx)

modification de shell32.dll :

<http://superuser.com/questions/494878/does-windows-8-have-a-status-bar-to-display-details-of-a-file>

idem

<http://sumtips.com/2012/11/windows-8-move-details-pane-to-bottom-of-explorer-window.html>

divers

<http://vistastylebuilder.com/forum/index.php?topic=215.0>

## **Premier écueil**

Mais si on tente d'ouvrir cette fenêtre en cliquant sur le lien «Arrière plan du bureau» ("Desktop Background"), on obtient l'erreur «ms-setting:personalization-background : le module spécifié est introuvable» ("ms-setting:personalization-background : the module is not find" ). En effet, ce lien pointe sur une application de l'interface Metro inexploitable dans WinPe.

Question : comment rendre actif ce lien ?

Après quelques recherches, en ouvrant le fichier themecpl.dll avec notepad, on trouve des chaînes de texte qui ressemblent à des scripts, à du xml. Elles contiennent les mots "duixml", "shellexecute", "pageWallpaper" et "pageColorization". En utilisant un logiciel explorant les ressources d'un programme, on retrouve les scripts dans les ressources. Une lecture permet de vite se faire une idée de la logique.

Le GUI de «Sons» ("Sounds") est lancé avec la commande :

Email : noelblanc.Winpe@free.fr



"shellexecute="%windir%\system32\control.exe" shellexecuteparams="/name Microsoft.Sound /page 2"

Le GUI de l'écran de veille est lancé avec :

"shellexecute=%windir%\system32\rundll32.exe" shellexecuteparams="shell32.dll,Control\_RunDLL desk.cpl,ScreenSaver,@ScreenSaver"

On trouve vite comment modifier le texte pour lancer les deux commandes qui nous intéressent.

La seule ressource à modifier : «UIFILE/1001»

Les deux chaînes de texte à modifier :

Avant	shellexecute="ms-settings:personalization-background"
Après	"shellexecute="%windir%\system32\control.exe" shellexecuteparams=" /name Microsoft.Personalization /page pageWallpaper"

Et

Avant	shellexecute="ms-settings:personalization-colors"
Après	"shellexecute="%windir%\system32\control.exe" shellexecuteparams=" /name Microsoft.Personalization /page pageColorization"

L'outil pour manipuler les ressources, explorer et modifier ( gros bouton vert de compilation ) et sauvegarder les changements : ResourcesHacker ici <http://www.angusj.com/resourcehacker/>

Il suffit de remplacer le fichier d'origine dans WinPe( le fichier mui est inchangé ).

Ensuite on teste ce nouveau fichier dans WinPe.

## **Second écran**

Le second écran est plus difficile à décrire sans copie d'écran. Mais cela allongerait trop le PDF.

La combobox "picture location" contient plusieurs entrées. Elles sont inopérantes dans mon contexte.

Seule "Solid Colors" est active. Je ne sais pas pourquoi.

Une solution :

- Lors de la construction de Winpe, je dépose les images "wallpaper" dans un répertoire à la racine de X, soit «x:\fleurs» dans mon cas. Comme j'utilise une VM, cela devient très simple. Pour un test sans VM, il faut monter le fichier Boot.Wim avec Dism par exemple, et copier le répertoire à la racine. Puis démonter.
- Dans le bureau de WinPe, j'ouvre le menu «Desktop Background» (faire un clic droit sur le fond d'écran et activer le lien "Desktop Background" ou lancer la bonne commande «control», voir ci-dessus).

- Cliquer sur le bouton «Parcourir...» ("browse...").
- Sélectionner le répertoire contenant les images. Soit «x:\fleurs» dans mon cas.
- Dans la combobox, sélectionner l'entrée correspondant au répertoire. Et les images apparaissent.

Vous pouvez alors changer le fond d'écran directement.

J'espère que quelqu'un va trouver pourquoi le fond d'écran initial n'apparaît pas, ni même son entrée correspondante dans la combobox.

Voir mon script «modify-themecpl.PS1»

## Win10pCap and Wireshark hors Boot.Wim : A REVOIR



Je fais le choix suivant : l'installation présentée ici est lancée après le démarrage de WinPe. Cela signifie que l'installation sera à refaire à chaque démarrage. C'est très rapide et on en a pas besoin tous les jours.

Mais il serait aussi possible, si les fichiers sont recopiés au bon endroit sur le média, de relancer le programme « installer.exe » qui installe le pilote « win10pcap » dans la ruche « system ». C'est ce que je fais dans ma VM.

On peut faire le choix d'installer ce pilote win10Pcap dans Boot.Wim, soit avec DISM, soit en modifiant la ruche « system » ( par exemple en modifiant le script traitement.ps1).

### ➤ Mon contexte :

une VM utilise un fichier VHD contenant WinPe en mode "flat".

L'installation pourrait être différente avec un autre contexte, par exemple WinPe contenu dans une clé usb

### ➤ Versions testées :

win10pcap 10.2.0.5002 version lue dans le fichier inf

wireshark 2.0.2

### ➤ Sites :

[www.wireshark.org](http://www.wireshark.org)

<http://www.win10pcap.org/>

## Win10pCap

L'installation affiche rapidement un message d'erreur « le répertoire d'installation doit être sur un disque dur local ».

Deux méthodes pour contourner :

- modifier le MSI avec ORCA
- ou extraire les fichiers (sys, inf, cat, etc) du fichier MSI avec 7z.

## Obtenir ORCA

Je l'ai trouvé dans le SDK de windows7 (64 bits) dont j'avais gardé une ISO.

Dans le fichier « E:\Setup\WinSDKTools\_amd64\cab1.cab » on trouve divers outils dont orca, signtools, makecert....

On ouvre ce fichier « cab1.cab » avec 7z. Et on extrait le fichier « WinSDK\_Orca\_Msi\_5E20C107\_DAA3\_4D49\_AFAE\_7FB2594F0CDC\_amd64 ».

On ajoute l'extension « .msi » à la fin de ce nom. On vérifie avec 7z qu'il contient bien une « structure msi ». On peut maintenant installer ce produit « orca.msi ».

## Méthode 1 : Modification et installation de win10pCap.Msi avec Orca

On crée une copie de sauvegarde du fichier « win10pCap.msi ».

On lance Orca et on ouvre le fichier « win10pCap.msi ». Dans la partie gauche « Tables », on sélectionne « ControlEvent ». Dans la partie droite, on sélectionne la ligne « installDirDlg » dont la colonne « Argument » contient « VerifyReadyDlg ». On double-clic sur le champs de la colonne « Condition » de cette ligne. On peut maintenant modifier la valeur. On remplace tout le contenu « WIXUI\_DONTVALIDATE....\_VALID='1' » par « 1 ». On valide. On sauvegarde.

Ensuite on installe ce fichier Msi dans WinPe (double-clic, ou ouvrir, ou ...)

## Méthode 2 : Extraction et installation du pilote win10pCap

Ouvrir le fichier « win10pCap.msi » avec 7Z.

Recherche à l'intérieur le fichier «win10pCap.cab ». Ouvrir ce fichier. Il contient les versions pour divers OS ( windows 7/8 et windows 10, en version 32 et 64 bits ). Extraire tous les fichiers dans un répertoire.

Et selon la configuration qui vous intéresse, copiez les trois fichiers (sys, inf, cat) dans un répertoire, ainsi que les deux fichiers wpcap.dll, paquet.dll, installer.exe.

Dans Winpe, on recrée l'arborescence suivante :

- « x:\program Files (x86)\win10pcap\x64\
  - wpcap.dll, paquet.dll, installer.exe
- « x:\program Files (x86)\win10pcap\x64\drivers/win10 »
  - win10pcap.inf, win10pcap.sys, win10pcap.cat
- se placer dans le répertoire « x:\program Files (x86)\win10pcap\ »
- lancer « installer.exe ».

## **Wireshark**

L'installation est simple : lancer le programme préalablement téléchargé depuis le site officiel.

Ne pas valider l'installation de WinPCap proposée par WireShark.

Il faut ajouter deux fichiers .dll suivants dans le répertoire x:\windows\system32 :

msvcp120.dll et msucr120.dll

Ces deux fichiers ne se trouvent pas dans la fourniture originelle de windows10Ent Build 10586. Je les prends sur ma machine sous windows10 hébergeant ma VM