Installing Windows 10X 195xx (emulator image) on real hardware

Here's some quick notes on installing Windows 10X on real hardware. For this example, we're assuming a system with *no* other critical disks installed, and a helpful host system being around to set up the initial image.

Prerequisites

Host

- Windows 10 Manganese build (195xx).
- Utility USB flash drive of ~32GB+.

Target

- CPU with Hyper-V support for VAIL.
- Graphics card with DCHU drivers available.
- UEFI system firmware with the ability to **disable Secure Boot**.
- Boot drive larger than 128 GiB. An 128 GB SSD usually isn't.
- Preferred: 4Kn boot drive. We'll provide steps a bit later for converting the image.

Host work

Fetch and mount the emulator image

Make sure you have a *clean* Flash.vhdx from the <u>W10X emulator</u>. Copy it someplace, and preferably keep another backup as well.

Mount it using PowerShell (as administrator):

Mount-VHD "X:\WCOS\Flash.vhdx"

Check if the emulator image is mounted correctly:

Get-StoragePool -FriendlyName OSPool

This should look like the following:

FriendlyName OperationalStatus HealthStatus IsPrimordial IsReadOnly Size AllocatedSize							
OSPool O	Ж	Healthy	False	False	127.9 GB	21.81	GB

Gather UpdateApp and verify it works

Start diskpart so you can mount MainOS:

list volume
select the volume called MainOS
select volume 42
assuming M: is free
assign letter=m
exit

From the MainOS partition, go and hunt down the following files and drop them in a standalone folder (for example, X:\WCOS\Tools):

\Windows\Servicing

- UpdateApp.exe
- CbsApi.dll
- CbsMsg.dll

\Windows\System32

- CbsCore.dll
- DrvServicing.dll

- IUSpaces.dll
- IUSpaces_vb.dll (copy and rename IUSpaces.dll)
- UpdateAPI.dll
- cimfs.dll
- cmiadapter.dll
- cmiaisupport.dll
- cmintegrator.dll
- dpx.dll
- drvstore.dll
- msdelta.dll
- mspatcha.dll
- mspatchc.dll
- turbostack.dll
- wcp.dll
- wdscore.dll

Run <u>cmd.exe</u> as administrator, go to the tool directory, and try getting the installed packages on the image:

cd /d X:\WCOS\Tools

updateapp getinstalledpackages

The result should look a lot like the following:

UpdateApp - Update Application for Windows Mobile [00:00:00] Loaded servicing stack from X:\wcos\tools with session name IUPackageInfoSession_EFIESP [00:00:00] External storage staging directory is: (null) [00:00:00] Closing session IUPackageInfoSession EFIESP [00:00:00] Loaded servicing stack from X:\wcos\tools with session name IUPackageInfoSession_MainOS [00:00:00] External storage staging directory is: (null) [00:00:01] Closing session IUPackageInfoSession MainOS 164 packages: Microsoft-OneCore-HyperV-Guest-UpdateOS-Package~31bf3856ad364e35~amd64~en-US~10.0.19563.1000, **UpdateOS** Microsoft-OneCore-HyperV-Guest-UpdateOS-Package~31bf3856ad364e35~amd6410.0.19563.1000, UpdateOS Microsoft-OneCore-ServicingStack-UpdateOS-Package~31bf3856ad364e35~amd6410.0.19563.1000, updateos Microsoft-OneCore-ServicingStack-UpdateOS-UX-Package~31bf3856ad364e35~amd6410.0.19563.1000,updateos Microsoft-OneCoreUpdateOS-Product-Package~31bf3856ad364e35~amd64~en-US~10.0.19563.1000, updateos Microsoft-OneCoreUpdateOS-Product-Package~31bf3856ad364e35~amd6410.0.19563.1000, updateos Microsoft-Windows-OneCoreUpdateOS-ImageCustomization-Package~31bf3856ad364e35~amd64 10.0.19563.1000, updateos

Microsoft-Composable-ModernPC-BootEnvironment-Core-CodeIntegrity-Sbcp-				
Package~31bf3856ad364e35~amd6410.0.19563.1000, EFIESP				
Microsoft-OneCore-BcdBootoption-Package~31bf3856ad364e35~amd64~~10.0.19563.1000, EFIESP				
[]				

getinstalledpackages completed successfully command took 7 seconds

If it does, congratulations! You can move on to the next step.

Inject graphics and network drivers

For this example we'll show the Intel HD Graphics driver, but you might need to add more INFs depending on your hardware. If you can't find the right INFs, why are you even doing this?

Place extracted Intel drivers in a directory, so that you have e.g.

X:\WCOS\DHCUDrivers\Graphics\iigd_dch.inf. Open iigd_dch.inf, and note down the values for 'Provider' and 'DriverVer'. For me, those were:

```
Provider=%Intel%
DriverVer=08/23/2019,26.20.100.7158
```

The provider name is an indirected variable here, so we go and find what <u>%Intel</u> meant as well. A bit below in the INF, we find the following:

Intel = "Intel Corporation"

Good! Now, invoke updateapp with the data we've just discovered to install the INF to the BSP partition in your WCOS image:

```
updateapp install "DriverPackage|X:\WCOS\DHCUDrivers\Graphics\iigd_dch.inf|Intel_Corporation-
iigd dch.inf~amd64~26.20.100.7158~bsp|0"
```

Note the recurrence of Intel_Corporation and 26.20.100.7518. The installation process will complain with an error code of c0880005 if you get the 'keyform' wrong.

The general rule for inf file names and provider names in the 'keyform' is the following:

- Any space in the inf name or the provider name must get replaced by an underscore '_'
- \bullet Any dash in the inf name or the provider name must get replaced by an underscore '_'

After you've installed your favorite driver packages, we can prepare the utility flash drive.

Make a utility flash drive

Gather the following assets into a directory we'll label X:\WCOS\UtilityDrive\Boot :

- From an ISO of Windows 10 19559 AMD64 (or higher see <u>UUPDump</u> or similar for generating these):
 - ∘ boot∖
 - $\circ \text{ EFI}$
 - sources\boot.wim
 - bootmgr.efi
- For later servicing, your WCOS\Tools folder. Use a hex editor to replace any mention of the Unicode string X:\Windows in UpdateAPI.dll and UpdateApp.exe with something like X:\Wbndows, or expect any servicing tasks to fail.
- An x64 <u>EFI shell</u>. Rename EFI\boot\bootx64.efi to EFI\boot\winx64.efi, and name the shell as EFI\boot\bootx64.efi. You'll need the shell in order to *ever* boot regular Windows again (including PE).
- A file called startup.nsh in the root:

dmpstore -d SecureBootPlatformID fs0:\efi\boot\winx64.efi fs1:\efi\boot\winx64.efi fs2:\efi\boot\winx64.efi fs3:\efi\boot\winx64.efi fs4:\efi\boot\winx64.efi fs5:\efi\boot\winx64.efi fs6:\efi\boot\winx64.efi fs8:\efi\boot\winx64.efi fs9:\efi\boot\winx64.efi fs9:\efi\boot\winx64.efi

If you are having troubles getting back to Windows PE/Windows Desktop, you may also try the following extra commands in startup.nsh: (**Warning**: these will kill every variables you have saved on your system)

dmpstore -d -guid BA57E015-65B3-4C3C-B274-659192F699E3 dmpstore -d -guid 77FA9ABD-0359-4D32-BD60-28F4E78F784B

- gdisk64.exe from GPT fdisk.
- ddrelease64.exe.

Partitioning

- 1. Connect your UFD.
- 2. Open diskpart.
- 3. list disk, select disk the right disk, or you'll lose all data on it and will have to do a long partition scan to have any hopes of retrieving your data, and clean + convert gpt.
- 4. create partition primary size=5000, format fs=fat32 quick, assign letter=y to make a bootable FAT32 partition.
- 5. create partition primary, format fs=exfat quick, assign letter=z to make an exFAT partition to house the VHDX.

Putting things in place

Place your boot drive directory on the drive you called Y: Dismount-VHD "X:\WCOS\Flash.vhdx in your PowerShell to unmount the VHDX, and copy the VHDX to Z: You should now have a tree structure similar to:

Y:\		
Boot\		
EFI\		
Sources\		
Tools\		
startup.nsh		
Z:		
Flash.vhdx		

Eject and unplug the UFD.

Target work

Use your throwaway laptop or other modern enough system with *larger-than-128GB* system drive. **Make sure Secure Boot is off.**

Boot Windows PE

Boot it on the target. Really. Once you get into Setup, press Shift-F10 to open a command prompt. Go back and open another, for good measure. Alt-Tab works for switching here.

Copy the VHD (destructive!)

Find out where your USB flash drive is mounted. This will involve doing a lot of the following:

C:
dir
D:
dir
E:
dir
F:
dir

Here, we'll assume the *boot* volume is D: and the volume with Flash.vhdx is E:.

Open diskpart, and attach the VHD:

select vdisk file=E:\flash.vhdx attach vdisk readonly # wait a minute or so list disk # if MainOS etc. show up as online, good!

Note down the ID of a 2048 MB disk with a 2048 MB free space, and subtract 1 from it.

note: there's no 16Disk 17 Online 2048 MB 2048 MB

The ID to note down, therefore, is 16. Also, note down the ID of the target disk (3 in this case).

Wipe it. Yes. That's data loss for you. Make sure you've got backups of anything important on there.

select disk 3 clean convert mbr (replacing 3)

Copy the VHDX's content to your disk:

E:\Tools\ddrelease64 if=\\.\physicaldrive16 of=\\.\physicaldrive3 bs=8M --progress

(replacing 16 and 3)

... and go have a hot beverage while waiting for this to hit 131072M.

Rebuild the GPT (for 512-byte disks only)

You probably have a 512-byte disk, so you're going to have to rebuild the GPT. Yay!

Run commands along the following:

> E:\tools\gdisk64 -I \\.\physicaldrive16 [..] Number Start (sector) End (sector) Size Code Name 1 512 8703 32.0 MiB EF00 BS_EFIESP 2 8704 33554426 128.0 GiB 4202 OSPool

Remember the numbers (start, end, code and name) for each partition. Multiply the numbers by 8 (since 4K/512 = 8) - so you get 4096, 69624, etc.

Now, we'll create a new GPT for the target disk:

```
E:\tools\gdisk64 \\.\physicaldrive3
# accept any warning
x
z
E:\tools\gdisk64 \.\physicaldrive3
```

accept the warning

n	
1	
4096	
69631	
EF00	
2	
69632	
268435415	
4202	
C	
1	
BS_EFIESP	
C	
2	
OSPool	
p	
check if it makes sense	
matches the above but	•

matches the above but with different sector numbers

W

Exit all open windows, and your system should reboot.

Boot Windows PE, again

Boot into Windows PE again - not the internal disk you just overwrote. Verify in diskpart if you can list volume and it'll show MainOS etc. without you having attached the VHD.

Remove WCOS Security

In Windows PE, open diskpart and do select volume. Find the volume named ÈFIESP we will assume here its volume id is 6, yours may be different. Then we run select volume 6 and assign. Do list volume again to find the drive letter of EFIESP, in our case it's E:, yours may be different.

Delete the following file: del E:\efi\Microsoft\Boot\SecureBootPolicy.p7b

You may additionally replace winsipolicy.p7b with the one from a desktop sku (the file is located in the same folder).

Boot W10X

OK, now you can boot your internal disk. If you haven't followed the Remove WCOS Security instructions, this will set a Secure Boot policy value, however, so you'll have to boot your utility flash drive again if you want to boot any other Windows media (or otherwise execute the dmpstore command).

If everything's right, you should be booting into Windows 10X, and your graphics adapter might even be working.

Revision #22 Created 12 February 2020 17:00:13 by nta Updated 24 November 2022 21:29:07 by Daniel Kornev