HSVE Hackintosh

Supported Hardware

Supported GPUs

Recommended brands to avoid: XFX, PowerColor, HIS and VisionTek. As they run a vBIOS that doesn't work well with macOS.
RTX and GTX 10xx/1xxx NOT supported. No support in macOS Sonoma and above for NVIDIA GPUs.

AMD RX 5xxx/6xxx Cards

- Radeon RX 6950 XT* (*Fake ID spoof required!)
- Radeon RX 6900 XT*
- Radeon RX 6800 XT
- Radeon RX 6800
- Radeon RX 6650 XT
- Radeon RX 6600 XT
- Radeon RX 6600
- Radeon RX 5700 XT
- Radeon RX 5700
- Radeon RX 5600 XT
- Radeon RX 5600
- Radeon RX 5500 XT
- Radeon RX 5500

Native AMD RX Cards

- Radeon Vega 64* (Temperamental, *Can sometimes be tricky to get fully working with PCI-Passthrough, might get issues with audio.)
- Radeon Vega 56*
- Radeon VII
- Radeon RX 590
- Radeon RX 580*
- Radeon RX 570
- Radeon RX 560X
- Radeon RX 560
- Radeon RX 550*
- Radeon RX 480
- Radeon RX 470D
- Radeon RX 470
- Radeon RX 460
- Radeon Pro WX 9100

- Radeon Pro WX 7100
- Radeon Pro WX 5100
- Radeon Pro WX 4100

Legacy Nvidia Cards

- Geforce GTX 780 Ti (Compatible)
- Geforce GTX 780
- Geforce GTX 770
- Geforce GTX 760 Ti
- Geforce GTX 760
- Geforce GT 740*
- Geforce GT 730*
- Geforce GT 720
- Geforce GT 710*
- Geforce GTX 690
- Geforce GTX 680
- Geforce GTX 670
- Geforce GTX 660 Ti
- Geforce GTX 660*
- Geforce GTX 650*
- Geforce GT 640*
- Geforce GT 635
- Geforce GT 630*
- Quadro Cards*

Supported WiFi/BT Cards

Most devices should work just fine with Monterey and Ventura. We can't confirm whether every single one will work 100%, but we do know for sure that there has been a change of drivers shipped with macOS Sonoma. Drivers for certain Broadcom devices has been deprecated, so do keep that in mind!

Notice

You yourself are responsible for which one of the adapters listed below. If the one you bought doesn't work as intended, there's nothing we can do or take any responsibility for!

Should you already have a built-in Wi-Fi/Bluetooth card installed on your motherboard and it's incompatible, make sure to remove it or disable in BIOS if possible. Will cause interference with any external cards installed.

Fenvi Cards

- Fenvi FV-HB1200 | Desktop PCI-E (Recommended)
- Fenvi T191 | Desktop PCI-E (Recommended)
- Fenvi BCM94360NG | M.2 Card
- Fenvi BCM94360CD | Desktop PCI-E
- WDXUN BCM943602CS | Desktop PCI-E
- Fenvi BCM94360NG | Antenna Kit
- WIRCARD BCM94360CS2 | M.2 Card
- BCM94360CS2 | M.2 Adapter
- BCM94360CS2/BCM943224PCIEBT2 | M.2 Adapter
- BCM943602CS BCM94360CS2 BCM943224PCIEBT2 BCM94331CD | M.2 to USB Header adapter
- Antenna Kits
- Built-in antenna

Compatible USB WiFi Adapters

Alfa AWUS036AC

Alfa AWUS036ACH
Archer T2U Plus (AC600)
Archer T2U NANO, MINI, AC600
Archer T3U
Archer T3U Plus
Archer T2U MINI V3
Archer T2U Plus
ArcherT4U V1, V2, V3
Archer T9UH V1, V2
ASUS USB AC68
ASUS USB-N13
ASUS USB Nano-AC53
BrosTrend FBA_AC3
COMFAST CF-811AC
COMFAST CF-812AC
Comfast CF-758F
Comfast CF-WU810N
Cudy WU1300S
Cudy WU700
Cudy WU650
CXFTEOXK
DLink DWA-121 N150
DLink DWA-131 E
DLink DWA-171 C
DLink DWA-182 D
DLink DWA-192 A
EDIMAX EW-7611UCB
EDIMAX EW-7722UTn V2
EDIMAX_EW-7822ULC
EDIMAX EW-7612Uan V2
EDIMAX EW-7833UAC
EDIMAX N300
EDIMAX EW-7811Un (N150)
EDUP EP-AC1689
Fenvi AC1300 (RTL8812bu)
FILOWA USB WiFi-RTL8812BU
Foktech AC600 Nano
Jensen Eagle 100-AC
Kextech MINI USB RTL8192
Linksys WUSB6300 V2
Linksys WUSB6400M
M-Tech UW-01 USB
Netgear A6100
Netgear A6150
Netgear A7000

Netis WF2120 N Nano USB
Plexgear AC1200
Sitecom WLA7100
TL-WN823Nv2/v3
TL-WN725Nv3
TL-WN723Nv2/v3
TL-WN722Nv2/v3
TL-WN821Nv6
TL-WN822Nv4/v5
TENDA W311-MINI
Tenda U3 Mini
TENDA U12
TRENDnet N150 Micro
TRENDnet TEW-808UBM
TRENDnet TEW-908UB
YUNCLOUD Realtek (RTL8814AU)
ZAPO W58L (RTL881lAU)

For assistance with hardware selection or Hackintosh builds, please contact us for consultation services.

Fixing Issues

Fixing Apple ID / iCloud Issues

This guide outlines how to fix issues with Apple ID not signing in / iCloud showing 'Verification Failed' and how to fix this for good!

Carry out the following steps if:

- You would like working Apple ID / iCloud On Your Hackintosh
- Are aware that no system updates will work when these patches are applied (Can be removed and re-added for system updates)
- Are aware that HSVE is not responsible if you get your Apple ID banned (Unlikely)

With OCAT GUI:

Go to Kernel -> Patch and add two new items as follows:

Identifier	Count	Find	Limit	Replace	Skip	Arch
kernel	1	68696265726 E6174656869 64726561647 90068696265 726E6174656 36F756E7400	0	68696265726 E6174656869 64726561647 90068765F76 6D6D5F70726 573656E7400	0	x86_64
kernel	1	626F6F74207 3657373696F 6E205555494 40068765F76 6D6D5F70726 573656E7400	0	626F6F74207 3657373696F 6E205555494 40068696265 726E6174656 36F756E7400	0	x86_64

Watch the following video along with these commands for better understanding! :

https://youtu.be/gp0Vi5OJO-k?si=DVF3PVRd9vJ-WHIC

Fixing High CPU Usage / Setting Custom CPU Cores

In this guide we outline how to fix the **High CPU Usage** and set **Custom CPU Core**s on your virtualized macOS VM using our updated method for Proxmox.

Important Information:

macOS only accepts CPU cores in **powers of 2**:

1, 2, 4, 8, 16, 32, 64, ...

We provide a workaround to use non-power-of-two core counts. This may or may not work for your system. If you encounter issues, simply revert the changes and ensure to take a backup!!

1. Editing your VM Configuration:

Open your VM Configuration file replacing [VMID] with your VM ID

```
sudo nano /etc/pve/qemu-server/[VMID].conf
```

Intel: Supported Core Configurations

Native (No Args Needed) These core counts work natively in macOS:
2 cores
4 cores
8 cores
16 cores
32 cores

Intel: 48 Core Examples

Two working options:

```
-smp 48,cores=2,sockets=24,threads=1,maxcpus=48
```

```
-smp 48,cores=2,sockets=12,threads=2,maxcpus=48
```

AMD CPUs

For AMD users, replace your current `-cpu` argument with:

```
args: <default-arg> -cpu host
```

Then simply set the desired number of CPU cores in **Proxmox → Hardware → CPU**.

If you need any further assistance please do not hesitate to contact us

Making Changes To Your Hackintosh VM

Resize Disk of your HSVE macOS VM

This guide has a video walkthrough, please consider watching the video here:
https://youtu.be/x9CokQvbFYc

Increasing Disk Size In VM Hardware Section - Proxmox Web UI

In the Hardware tab of your VM, select your disk (e.g., `virtio0`) and click 'Disk Actions' at the top.

Enter the 'amount you want to increase it by', *not the new total size*.

Example: If your current disk is '250GB' and you want '500GB total', you must type '250'.

Expanding Disk In macOS After Increasing In VM Hardware

Boot into your macOS VM, open 'Terminal', and run:

```
diskutil list
```

This will show your disks. Example: `disk0` might be '524.4GB', but the APFS container (`disk3`) is still only '255.8GB'.

Your disk numbers may be different — double check before running commands!

Repair the Disk Partition Run:

```
diskutil repairDisk disk0
```

Type 'y' when prompted.

4Resize the APFS Container Run:

```
diskutil apfs resizeContainer disk3 0
```

This expands the APFS volume to use all available space.

Verify TRIM Status Check in 'System Information → SATA/NVMe' if 'TRIM' is enabled.
If disabled, run:

```
sudo trimforce enable
```

Done! Your macOS VM should now be using the newly expanded disk space.

If you require any support please contact us

RAW vs. QCOW2 Making Changes To Your VM

RAW Disk Image

Raw images offer a “raw” characteristic that results in performance comparable to physical drives, making them highly efficient. Moreover, this feature allows for direct attachment to VMs. Another advantage of raw images is their straightforward conversion to various image types.

Raw signifies being in its original and unformatted form, such as a disk. In Linux, a raw image is a type of unadulterated binary image. In a file system that allows sparse files, a raw image only occupies the real storage capacity of the disk data.

qcow2 Disk Image

qcow2, also known as QEMU Copy on Write 2, is a virtual image format that is compatible with the QEMU emulator. It is a popular choice in virtual environments, similar to raw image, and currently offers nearly identical performance.

Benefits over raw Disk Image

The qcow2 image requires less storage space because it does not support holes in the file system. In contrast, the raw image is typically larger in size. The qcow2 file only expands when the virtual machine utilizes the disk space, resulting in a more efficient storage solution. With COW and copy-on-write, the qcow2 image reflects only the modifications made to the underlying disk. Snapshots are supported by qcow2 images, and it is possible to have multiple snapshots in one image. Within the qcow2 file format, zlib compression can be applied independently to each cluster, providing enhanced compression capabilities. The support for AES encryption in qcow2 allows for encryption using a 128-bit key. Converting a Disk Image Format Should you already have a Virtual Machine running on Proxmox with one or the other format and want to switch for the benefits over the other, you can convert using the QEMU-img conversion tool, which comes bundled in Proxmox.

Converting qcow2 to raw

```
qemu-img convert -p -f qcow2 -O raw /<folder>/<disk-image>.qcow2 /<folder>/<disk-image>.raw
```

Converting raw to qcow2

```
qemu-img convert -p -f raw -O qcow2 /<folder>/<disk-image>.raw /<folder>/<disk-image>.qcow2
```

How To Passthrough GPU To Your VM

PCI Device passthrough does allow you to pass over things like your GPU for proper acceleration within your Virtual Machines, which makes it possible to run games and other graphic intensive tasks with no performance loss compared to your bare-metal installed Windows for example.

A single guide cannot possibly cover all the different systems that exist. Our guide attempts to give you as much information as possible. There is a lot of information out there, and not all of it may be relevant to you and your system, do be sure to make some research of your own.

Configuring the system BIOS Important BIOS settings for getting things working properly!

- Make sure your BIOS is on it's latest update.
- Enable SVM/Virtualization Mode
- Enable VT-d
- Required for passthrough to work at all; supported CPUs.
- Enable SR-IOV
- If present.
- Disable CSM/Legacy Boot
- Enable ACS/PCI-E Access Control
- If present, set to Enabled (Auto doesn't work).
- Set PEG {NUMBER} ASPM to L0 or L0sL1
- Set whichever amount you've got to mentioned, if L0 is present, pick that.
- Enable 4G Decoding
- Disable Resizable BAR/Smart Access Memory
- You might experience 'Code 43 error' if enabled.
- Enable IOMMU
- If present, mostly for AMD boards; supported CPUs.
- Set Primary Display to CPU/iGPU
- If your CPU has an iGPU, if not skip this.
- Set Pre-Allocated Memory to 64M
- If your CPU has an iGPU, if not skip this too.
- This step is not necessary for the passthrough, but helps keeping things clean.

For ignoring some annoying "warnings" in your dmesg output, do the following

```
nano /etc/modprobe.d/kvm.conf
options kvm ignore_msrs=Y report_ignored_msrs=0
```

Hit Ctrl + X -> Y -> Enter to save changes.

Setting the Boot arguments For Intel CPUs

For GRUB; Replace the current similar line with this one in your grub file.

```
nano /etc/default/grub
```

Make these changes to that file.

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet intel_iommu=on iommu=pt initcall_blacklist=sysfb_init"
```

Hit Ctrl + X -> Y -> Enter to save changes.

For AMD CPUs

For GRUB; Replace the current similar line with this one in your grub file.

```
nano /etc/default/grub
```

Make these changes to that file.

```
GRUB_CMDLINE_LINUX_DEFAULT="quiet iommu=pt initcall_blacklist=sysfb_init"
```

Hit Ctrl + X -> Y -> Enter to save changes.

Your system might not be relying on Grub, but systemd instead, this is the case when you're using ZFS. So for systemd;

```
nano /etc/kernel/cmdline
root=ZFS=rpool/ROOT/pve-1 boot=zfs intel_iommu=on iommu=pt initcall_blacklist=sysfb_init
```

Hit Ctrl + X -> Y -> Enter to save changes.

Your system might not be relying on Grub, but systemd instead, this is the case when you're using ZFS. So for systemd;

```
nano /etc/kernel/cmdline
root=ZFS=rpool/ROOT/pve-1 boot=zfs iommu=pt initcall_blacklist=sysfb_init
```

Hit Ctrl + X -> Y -> Enter to save changes.

For GRUB only Only if your system uses Grub, always required after making changes!

```
update-grub
```

For systemd only Only if your system uses systemd, always required after making changes!

```
pve-efiboot-tool refresh
```

Reboot to commit changes

```
reboot
```

Configuring IOMMU Once the Host is up and running again, we need to verify that IOMMU is enabled

For Intel CPUs

```
dmesg | grep -e DMAR -e IOMMU
```

For AMD CPUs

```
dmesg | grep -e DMAR -e IOMMU -e AMD-Vi
```

If there is no output, something is wrong. You should be seeing something like this; "DMAR: IOMMU enabled"

Enable necessary kernel modules, run the following command

```
nano /etc/modules
```

Add the following lines;

```
vfio
vfio_iommu_type1
vfio_pci
```

Hit Ctrl + X -> Y -> Enter to save changes.

After changing anything modules related, you need to refresh your initramfs.

```
update-initramfs -u -k all
```

Now check if remapping has been enabled.

```
dmesg | grep remapping
```

Should output something like this; For AMD CPUs "AMD-Vi: Interrupt remapping enabled" For Intel CPUs "DMAR-IR: Enabled IRQ remapping in x2apic mode" x2apic can be different on older CPUs, but should still work. Only if you encounter issues, consider this; You could also try adding in nox2apic to your Grub or Systemd args.

If your system doesn't support interrupt remapping, you might want to allow unsafe interrupts. Please be aware that this option might make your system unstable, but not necessarily.

```
nano /etc/modprobe.d/iommu_unsafe_interrupts.conf
```

Add the following line;

```
options vfio_iommu_type1 allow_unsafe_interrupts=1
```

Hit Ctrl + X -> Y -> Enter to save changes.

Blacklisting driver modules to give the VMs full access to the graphics cards etc.

```
nano /etc/modprobe.d/pve-blacklist.conf
```

Add the following lines;

```
blacklist nouveau
blacklist nvidia
blacklist nvidiafb
blacklist snd_hda_codec_hdmi
blacklist snd_hda_intel
```

```
blacklist snd_hda_codec
blacklist snd_hda_core
blacklist radeon
blacklist amdgpu
```

Hit Ctrl + X -> Y -> Enter to save changes.

For PCI passthrough to work properly, you need to have a dedicated IOMMU group for each of the devices you want to assign to a VM. To make sure that's the case, do the following;

```
pvesh get /nodes/{nodename}/hardware/pci --pci-class-blacklist ""
```

Replace {nodename} with the name of your Proxmox node.

This is an example of one of my nodes. As you can see, my GPUs, USB Controllers, Wireless adapter has their own dedicated group.

If your devices are in the same group, be sure to double check for ACS (Access Control Service) in your BIOS. Should you not find it anywhere, you can apply an override patch by including pcie_acs_override=downstream,multifunction in your grub or systemd file, if so go back to the step where we edit our said system file.

Blacklisting your Devices Finding the corresponding IDs for your PCI devices, run the following command

```
lspci -nn | grep -i {device}
```

{device} = vga, nvidia, usb, audio, wireless, and so on. Don't include the { } brackets.

You should then see a list similar to below; "0x.00.x USB controller ... [1234:5678]" "0x:00.x VGA compatible controller ... [1234:5678]" "0x.00.x Audio Device ... [1234:5678]" Not necessary to take note of the GPU Audio ID. Only do one device reference at the time, and take note of the IDs you need.

As you can see here, the ids for my GPUs are 10de:1287 and 1002:67d. You might want to look for your USB Controller ids as well, if you encounter any issues with USB Passthrough down the line. Keep yours noted for the next step!

In general it's recommended to add the IDs for all devices you plan on using for a VM. What the blacklisting essentially does, is to block the devices for Proxmox holding them hostage, and not letting a VM have full access to it, at least in most cases.

Blacklisting the PCI device IDs for the host, run the following command

```
nano /etc/modprobe.d/vfio-pci.conf
```

Add the Device IDs in this file like this;

```
options vfio-pci ids=1234:5678,1234:5678 disable_vga=1
```

Hit Ctrl + X -> Y -> Enter to save changes.

Again, do not add the GPU Audio ID. Note that adding disable_vga=1 here will probably prevent guests from booting in SeaBIOS mode, if you use that for anything.

You can also add disable_idle_d3=1 to the end of the line, essentially disables the D3 Device Power State; which will prevent certain hardware to enter a low-power mode, that might cause issues when some are passed through, read more here. Does no harm with disabling it, might only consume more power down the line. Has shown better stability for Thunderbolt cards for example.

Now when you got all that sorted, you want to reboot your Host machine once again.

```
reboot
```

Adding your Devices to your VM(s) When your Host is up and running again, we can finally start adding in our accessories to our VMs! Before you continue, depending on your choice of GPU, AMD or Nvidia, do take a look at the bottom of the page first for making necessary configurations to your macOS.

This is totally normal with the Console window whenever a GPU is added or Display is set to none.

So, make sure you got your monitor hooked up or at least a dummy plug so that you are able to remote into the VM, with your preferred Remote App.

This is the setting you want to have set for your GPU to fully work. For some, only All Functions and ROM-Bar can be ticked. Do your own experimenting with these options!

Some cards also requires a dumped .rom file for it to work. To acquire that you must dump the file yourself from your specific card, look for the guide further down in this post.

We recently discovered that if PCI-Express is not ticked, things like HDMI-Audio might not work. But do note that your install might not boot with it ticked, then have it un-ticked.

vBIOS Dumping (optional) AMD Vendor Reset (optional) Some tweaks might be necessary depending on the type of GPU you've got, you'll need mount your EFI disk and open this file in your preferred OpenCore config editor.

AMD Users EFI/OC/config.plist

Mount your EFI drive and open the file in OpenCore Configurator. Boot-args goes under NVRAM -> 7C436110-AB2A-4BBB-A880-FE41995C9F82

AMD RX 5000 and 6000 series requires the following boot-arg to get proper output. agdpmod=pikera

Only use OpenCore Legacy Patcher if you got any GTX 600 or 700 series by Nvidia, as that series is the only ones compatible with macOS.

AMD RX 400/500 and RX 5000/6000 series does not require any patching, as they are native cards.

Nvidia Users EFI/OC/config.plist boot-args goes under NVRAM -> 7C436110-AB2A-4BBB-A880-FE41995C9F82 amfi=0x80 AMFI is enabled ngfxcompat=1 Force compat property missing ngfxgl=1 Force OpenGL property missing nvda_drv_vrl=1 nvda_drv(_vrl) variable missing

To solve the SIP error, change csr-active-config to 030A0000 Upon reboot, select Reset NVRAM from the boot-picker.

If you require any assistance with any of the above please contact us.