

A Power User's Guide to VMware Fusion

Disclaimer: This is a personal document and is not official or endorsed by VMware. Feedback, suggestions, and edits are welcome.

This document is primarily intended for people who have used Fusion for a while and are curious about how to do more advanced things. People who have used other VMware products may be interested in this as a reference for where to find settings. This document describes configurations which have no UI settings and are **not supported**, but are still useful. **This document assumes you are familiar with [A Beginner's Guide to VMware Fusion](#).**

Important: Whenever you do file operations (move, copy, edit, delete, etc.) to a VM, *make sure it is powered down and Fusion isn't running*. You don't want to change the data out from under Fusion.

If you want to be notified of changes and additions to this document, you can use the "Receive email notifications" action in the sidebar on the left. Please use the comments below only for things specific to this document; general questions are better off in the [discussion section](#).

A tool you may find handy is [VMX Extras](#), which is a GUI way to change some of these settings.

Fit Full Screen

Note: This setting is no longer necessary in Fusion 2.0, as this is the default.

When you switch to fullscreen mode in a guest without tools installed or a guest which changes the screen resolution, you might notice black borders around the screen since the host resolution remains the same. You can tell Fusion to scale the guest to the host's resolution by editing `/Users/yournamehere/Library/Preferences/VMware Fusion/preferences` to include the line

```
pref.autoFitFullScreen =  
"fitHostToGuest"
```

If the preferences file doesn't exist, create it as a plain text file. If the `autoFitFullScreen` line exists, replace it or comment out the old line.

Boot Delay

Fusion's BIOS flashes by very quickly - this is good for normal use (optimize the common case!) but annoying if you want to change a BIOS setting. You can slow down the boot process by adding the following line to the `.vmx`:

```
bios.bootDelay = "3000"
```

You can of course change this number; the units are on the order of milliseconds (e.g. "3000" adds approximately 3 seconds delay).

In Fusion 3.0, an easier way to change the boot order (the most common reason you'd want to get into the BIOS) is via `Virtual Machine > Settings > Advanced > Startup Device`.

Scripting

Fusion 2.0 introduces `vmrun`, a way to interact with virtual machines through the command line. It's located in `/Library/Application Support/VMware Fusion/vmrun`; run it with no arguments for help. An example of some Automator actions which use `vmrun` is in [Automator Actions](#).

`vmrun` does not exist in Fusion 1.x, but you can get some similar functionality by using Applescript's UI scripting, for example like [Re: Fusion's handling of Fullscreen is driving me INSANE!](#).

Additionally, you can tell Fusion to take action on certain signals by adding either (or both) option to the `.vmx` (for just that VM) or `~/Library/Preferences/VMware Fusion/config` (for all your VMs):

```
signal.suspendOnHUP = "TRUE"  
signal.powerOffOnTERM = "TRUE"
```

You might also be interested in the guest power scripts, which depend on Tools being installed and are run inside the guest when it powers on/powers off/suspends/wakes from sleep. To see where these scripts are, open the Tools and select the Scripts tab.

Headless Mode

As the Beginner's Guide said, the `vmware-vmx` process does the real work of running the virtual machine while the Fusion UI process handles input and drawing. If you don't need the UI process, you can kill the UI process after you start the VM (e.g. `ctrl-option-clicking` the Fusion Dock icon and Force Quitting). The `vmware-vmx` process should continue to run in the background. You can reconnect to it by starting Fusion again and opening the VM.

In Fusion 2.0, an easier way is to run the following command in a Terminal Window:

```
defaults write com.vmware.fusion  
fluxCapacitor -bool YES
```

This will add a View menu item, "Headless". If you use this option, you probably should also use the signal config options mentioned in the Scripting section of this document to allow you to safely shut down the physical machine without having to reconnect to the virtual machine.

I believe the VM continues running even if you log out, since the `vmware-vmx` process is root-owned.

The `fluxCapacitor` option was removed from Fusion 3.0 due to rearchitecting of the rendering engine; we did not have time to make sure that headless mode still worked. We realize it's something that some people find useful. In the meantime, force quitting the UI or invoking Fusion directly should work.

Networking

[Dave Parsons](#) has written a good guide to custom network settings: [How to modify Fusion network settings whitepaper](#), as well as scripts to manage custom settings: [Advanced Networking Configuration - Tokamak Networking Scripts for VMware Fusion](#)

Share Guest Internet Connection With Host

WoodyZ has written a good guide for setting this up: [Share Windows XP Guest Internet Connection with OS X Host HOWTO](#)

Static IP address

This only applies to NAT and host-only modes; bridged mode does not involve Fusion's DHCP server.

You might want a particular guest to always receive the same IP address via Fusion's DHCP server: [Re: DHCP reservations please](#)

Alternately, by default Fusion's DHCP server reserves the range x.y.z.3-x.y.z.127 (where x.y.z is the vmnet1 or vmnet8 subnet) for static IPs - simply set the guest to use a static address in this range (the exact method to do this will depend guest-specific).

Arbitrary MAC address

Fusion 2.0 allows you to use arbitrary MAC addresses, not just the range assigned to VMware. In addition to changing the MAC address in the .vmx file to the one you want, you also need to add the following line:

```
ethernet0.checkMACAddress =  
"FALSE"
```

This is easier in Fusion 3.0. You should be able to set the MAC address via Virtual Machine > Settings > Network > Advanced options.

VNC Server

Like Workstation, Fusion has a built-in VNC server. This allows you to connect to the guest without having a VNC server installed in the guest - useful if a server doesn't exist for the guest or if you need access some time when a server would not work (say during the boot process). It's also good in conjunction with Headless Mode.

The VNC server is set up on a per-VM basis, and is disabled by default. To enable it, add the following lines to the .vmx:

```
RemoteDisplay.vnc.enabled =  
"TRUE"  
RemoteDisplay.vnc.port = "5901"
```

You can set a password with RemoteDisplay.vnc.key; details for how to calculate the obfuscated value given a plaintext password are in [Compute hashed password for use with RemoteDisplay.vnc.key](#).

Note Previous revisions of this document incorrectly mentioned that you could use a plaintext RemoteDisplay.vnc.password. This option was in previous versions of Workstation but got removed in favor of RemoteDisplay.vnc.key; it was not added back to Fusion until 2.0. The obfuscated RemoteDisplay.vnc.key is preferred.

If you want more than one VM set up in this manner, make sure they have unique port numbers.

To connect, use a VNC client pointing at *host-ip-address:port*. If you connect from a different computer, you may have to open a hole in the OS X firewall. If you use Leopard's Screen Sharing.app on the same computer as Fusion, don't use port 5900 since Screen Sharing refuses to connect to that.

This is easier in Fusion 3.0; VNC settings are under Virtual Machine > Settings > Advanced > Other.

Swap Alt (a.k.a. Option) and Windows (a.k.a. Command) keys

Mac and PC keyboards have different positions for the Alt/Option and Windows/Command keys. If you're using a PC keyboard or have years of PC muscle-memory, you can tell Fusion to swap these keys by adding the following line to `~/Library/Preferences/VMware Fusion/config`

```
mks.keyboard.swapAlt = "TRUE"
```

If the config file doesn't exist, create it as a plain text file.

This may not work in Fusion 2.0b2 and later, because Fusion's keyboard mappings have changed. Instead, set this mapping Fusion's Preferences.

USB HID

You might have noticed that some USB devices, such as keyboards and mice, do not show up in the Virtual Machine menu or in the status bar. By default, Fusion screens out Human Interface Devices (HID) because if you attach your mouse/keyboard to a VM, you may have no way to get back out (shutting down the guest may work, and if you've configured Fusion to not automatically connect, unplugging/replugging should work, but if you don't have this set and the guest crashes, you're out of luck). A similar problem exists if you use a Bluetooth mouse/keyboard and attach the Bluetooth adapter to the guest - don't do that either!

This masking is a problem for other devices such as tablets or mice with lots of buttons, since they are also HID and therefore don't appear in the list. If you want to use pressure sensitivity or other advanced features, you need to attach the device to the guest. To get HID entries to appear, add the following line to the `.vmx`:

```
usb.generic.allowHID = "TRUE"
```

If you use this setting, I would recommend disabling Virtual Machine > Settings > USB > Automatically Connect USB Devices so that you have some way to get a mis-connected device back (unplug and replug). Remember not to connect your only keyboard/mouse to the guest!

Example instructions for using a tablet with an Ubuntu guest: [Instructions: enable wacom pressure sensitive tablet in Ubuntu 7.10, Leopard & VMware Fusion 1.1](#)

"Two" computers in one

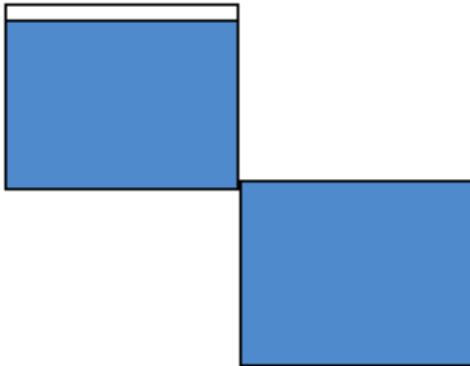
By now you know that Fusion lets you run multiple OSes at once, but it's limited because only one person can be using them at a time. What if you could separate them even more so that the host and the guest could be used by different people simultaneously? You can combine the USB HID setting with software cursor rendering to do this! While you could achieve a similar effect by combining VNC/headless mode, this method does not require additional physical computers as clients. Major caveats include:

- The host is still in control (so if OS X goes to sleep, the virtual computer will not be usable, Exposé will affect the guest window, etc.)
- You will need enough hardware resources to handle both host and guest
- The guest cursor isn't quite as smooth as normal (but is still quite usable)

To use this tip, you'll need a second keyboard/mouse for the guest and preferably have a second monitor. I would recommend disabling Virtual Machine > Settings > USB > Automatically Connect USB Devices so that you have some way to get a mis-connected device back (unplug and replug). Start by adding the following lines to the guest:

```
svga.noHWCursor = "TRUE"  
usb.generic.allowHID = "TRUE"
```

Connect the second keyboard/mouse to the guest, and optionally move the guest to the secondary monitor and go fullscreen. Since the host mouse can still wander over to the guest (get back out with ctrl-cmd on the host keyboard), if you use a second monitor, I suggest telling OS X that the monitors are offset so it's harder to do, e.g. go to System Preferences > Displays > Arrangement and drag the displays so they are arranged as follows:



You can extend this for even more virtual computers, though your hardware requirements will go up too.

Paravirtualization

Note: This setting provides no benefit on Nehalem or newer processors. The benefit of VMI is simpler page mapping, but Nehalem processors implement this efficiently in hardware.

VMware proposed the [VMI specification](#) as a vendor-independent method for hypervisors to talk to guests about certain things common to all hypervisors. VMI is part of the standard Linux kernel since 2.6.22, though it might not be enabled by default in your distro of choice. Enabling VMI [speeds up guest system calls, improves timekeeping, and reduces overhead](#) - of course, improvement varies by workload.

To do this, add the following line to the .vmx:

```
vmi.present = "TRUE"
```

For example, with 32-bit Ubuntu 7.04, if you've done this correctly, the dmesg output should contain the line "Booting paravirtualized kernel on vmi" instead of "Booting paravirtualized kernel on bare hardware"

Note that VMI does not work with 64-bit guests. On newer machines, it is also unlikely to help, as the pain point it was created to address has been eliminated due to improvements in hardware-assisted virtualization.

USB Logs

You might remember from [Information Gathering for VMware Fusion](#) that Fusion is able to log USB traffic. We've released an open source (and unsupported) tool for visualizing USB logs generated in this manner: <http://vusb-analyzer.sourceforge.net/>

GDB Server

Like Workstation, Fusion has a built-in GDB server. This allows you to debug the guest without having to run kdb or recompile your kernel (I think this is for Linux guests, or at least guests you can use GDB with). You will need a kernel with symbols for your distro and be familiar with how to use gdb. There is also untested support for record/replay, which lets you do interesting things like stepping **backward** through a program.

Depending on the guest, add the appropriate line to your .vmx:

32-bit guests	64-bit guests
 debugStub.listen.guest32 = "TRUE"	
 debugStub.listen.guest64 = "TRUE"	

32-bit guests	64-bit guests

Once you've gotten to the guest to where you want to debug, use gdb to connect from the host (or other computer) with

32-bit guests	64-bit guests
 target remote yourmachost:8832	
 target remote yourmachost:8864	

32-bit guests	64-bit guests

Note *yourmachost* can be localhost.

In GDB, load up the kernel with symbols:

```
file yourkernelfilehere
```

(substituting the appropriate file, e.g. vmlinux-2.4.21-27.EL.debug) and you're ready to debug the kernel. Debugging applications takes a little more work, see the following links for more details:

- <http://stackframe.blogspot.com/2007/04/debugging-linux-kernels-with.html>
- <http://stackframe.blogspot.com/2007/04/workstation-60-and-death-of.html>
- <http://stackframe.blogspot.com/2007/09/application-debugging-with-recordreplay.html>

<http://stackframe.blogspot.com/2007/10/configuring-application-debugging-with.html>