

Adding a Network Card to CentOS Linux

Detect & Configure The New Network Adapter

1. Determine existing network interfaces

```
ifconfig -a
```

2. Change directory to the network scripts folder

```
cd /etc/sysconfig/network-scripts
```

3. Clone the existing eth0 device network script

```
cp ifcfg-eth0 ifcfg-eth1 # this assumes the old card was eth0 and the new one is eth1
```

4. Get the Hardware Address for the eth1 network card, again this assumes the new card is eth1

```
grep eth1 /etc/udev/rules.d/70-persistent-net.rules
```

```
#you can get fancy and use awk and cut to isolate the string containing the Hardware Address
```

```
grep eth1 /etc/udev/rules.d/70-persistent-net.rules | awk -F"," '{print $4}' | cut -d= -f3
```

5. Replace all occurrences of eth0 with eth1 in the new network configuration script

```
sed -i 's/eth0/eth1/g' ifcfg-eth1 # or edit it by hand and change eth0 to eth1 where it appears
```

6. Edit the eth1 network configuration script and replace the Hardware Address with the one in the 70-persistent-net.rules file

```
vi ifcfg-eth1
```

7. Bring the eth1 interface up

```
ifup eth1
```

CentOS 6 64-bit on ESXi 5.0 – No Network After Cloning

Scenario: Installed CentOS 6.x x86_64 on VMWare ESXI 5.5

Upon login, noticed no NICs detected

see: {Centos 6.2 set static IP@<https://www.linuxquestions.org/questions/linux-server-73/centos-6-2-set-static-ip-931188/>}

see: {Setting up a new network device in CentOS@<http://linuxtoolkit.blogspot.com/2013/09/setting-up-new-network-device-in-centos.html>}

Regenerate the Persistent Network Rules Using udevadm

```
[code language=""]
#Regenerate the file using udevadm
rm -f /etc/udev/rules.d/70-persistent-net.rules
udevadm trigger --action=add
reboot
[/code]
```

Manually Verify and Correct NIC Device

```
[root@localhost ~]# ifconfig
lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 b)  TX bytes:0 (0.0 b)

[root@localhost ~]# dmesg | grep -i vmx
VMware vmxnet3 virtual NIC driver - version 1.1.30.0-k-NAPI
vmxnet3 0000:03:00.0: # of Tx queues : 1, # of Rx queues : 1
vmxnet3 0000:03:00.0: PCI INT A -> GSI 18 (level, low) -> IRQ 18
vmxnet3 0000:03:00.0: setting latency timer to 64
vmxnet3 0000:03:00.0: irq 56 for MSI/MSI-X
vmxnet3 0000:03:00.0: irq 57 for MSI/MSI-X
vmxnet3 0000:03:00.0: eth0: NIC Link is Up 10000 Mbps

[root@localhost ~]# _

[root@localhost network-scripts]# lspci | grep -i ethernet
03:00.0 Ethernet controller: VMware VMXNET3 Ethernet Controller (rev 01)

[root@localhost network-scripts]# _

[root@localhost network-scripts]# vi /etc/sysconfig/network-scripts/ifcfg-eth0_
DEVICE=eth0
BOOTPROTO="dhcp"
ONBOOT="yes"

[root@localhost network-scripts]# system-config-network_
```

```
[code language="bash"]
#1. Verify problem (no ethNn present)
ifconfig
#2. Verify system detects ethernet hardware
dmesg | grep -e 'vmx\|eth'
#3. Verify ethernet driver is loaded
lspci | grep -i ethernet
#4. If not already present, create generic eth0 config file
vi /etc/sysconfig/network-scripts/ifcfg-eth0
#If there is already a UUID line, Delete it, save, exit
#5. If this system was cloned, you'll need to remove the old
MAC address reference.
#See next step
[/code]
```

Manually Correct NIC MAC Address After Cloning

```
vi /etc/udev/rules.d/70-persistent-net.rules
# This file was automatically generated by the /lib/udev/write_net_rules
# program, run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single
# line, and change only the value of the NAME= key.
# PCI device 0x8086:0x100f (e1000) (custom name provided by external tool)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR(address)=="00:12:34:56:78:a
d", ATTR(type)=="1", KERNEL=="eth*", NAME="eth0"
# PCI device 0x8086:0x100f (e1000)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR(address)=="00:12:56:78:90:d
e", ATTR(type)=="1", KERNEL=="eth*", NAME="eth1"
# This file was automatically generated by the /lib/udev/write_net_rules
# program, run by the persistent-net-generator.rules rules file.
#
# You can modify it, as long as you keep each rule on a single
# line, and change only the value of the NAME= key.
# PCI device 0x8086:0x100f (e1000) (custom name provided by external tool)
SUBSYSTEM=="net", ACTION=="add", DRIVERS=="?*", ATTR(address)=="00:12:56:78:90:d
e", ATTR(type)=="1", KERNEL=="eth*", NAME="eth0"
"/etc/udev/rules.d/70-persistent-net.rules" 8L, 460C written
reboot
```

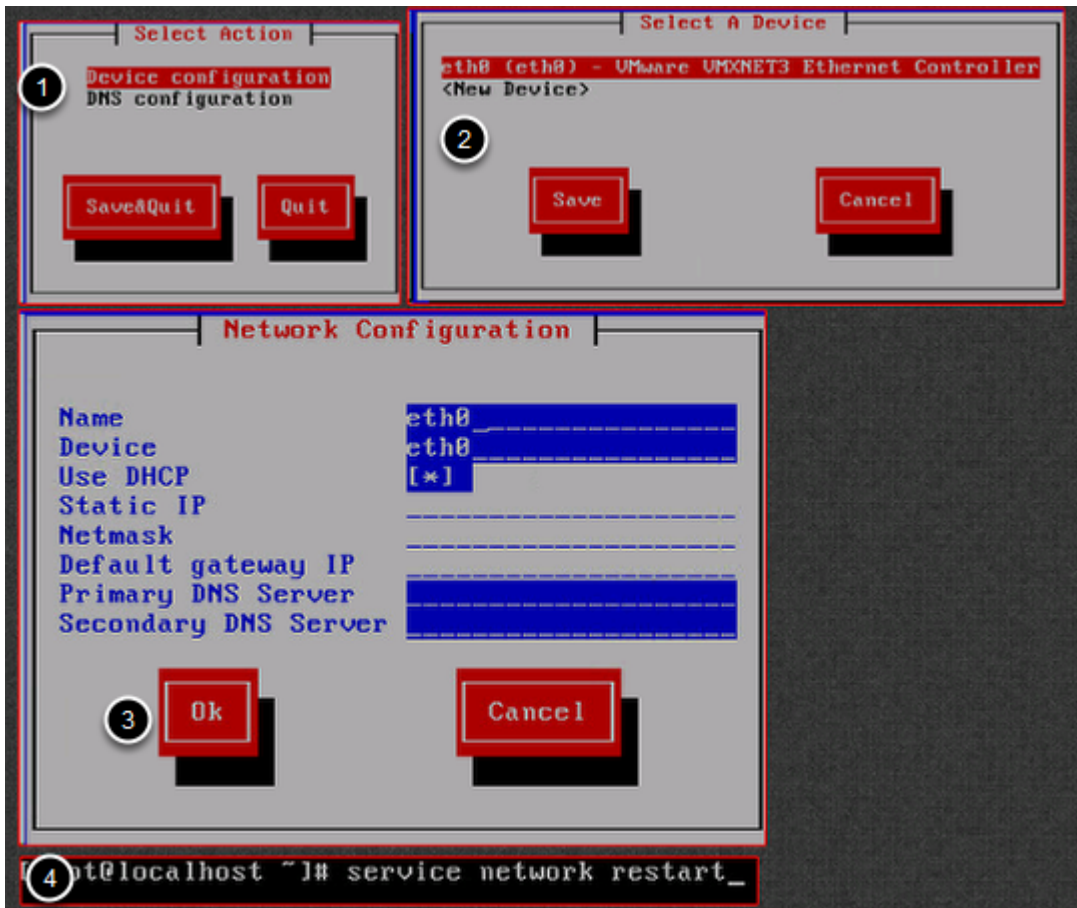
After cloning a CentOS 6.x machine, you may find that connectivity is limited to only the loopback (127.x.x.x) interface

This is likely because the system has retained the MAC address specification of the machine from which it was cloned

You can correct this by modifying the Persistent Network Rules File

```
[code language="bash"]
vi /etc/udev/rules.d/70-persistent-net.rules
[/code]
```

Configure NIC Device Using System Config Network Utility



1. Launch the System Network Configuration Utility

```
[code language="bash"]
system-config-network
[/code]
```

2. Choose 'Device configuration' in the dialog

3. Fill in settings accordingly (e.g. Name:eth0;Device:eth0;DHCP:enabled. Press OK

You'll be taken back to the original dialog

4. Press Save&Quit. Restart networking services

Settings should persist even after reboot