# **Bare Metal Installation of Windows Server using Razor**

# **by Lee Turchin**

**Goal:** This guide shows how to set up and use a customized version of Razor for unattended OS installs on Windows Server 2016 utilizing PXE booting. We will achieve a bare metal installation of Windows Server 2016 using Razor.

**References:** [**https://github.com/puppetlabs/razor-server/wiki/Installing-windows**](https://github.com/puppetlabs/razor-server/wiki/Installing-windows)

[**BM Deployment with Razor (4th Ed.)**](http://robroy.sie.hds.com/puma/bm/razor4/#_installing_code_jq_code) **-** Rob Roy’s document

[**Windows ADK for Windows 10 scenarios for IT Pros**](https://docs.microsoft.com/en-us/windows/deployment/windows-adk-scenarios-for-it-pros)

[**Bare Metal Provisioning API V7.1.0**](https://teamsites.sharepoint.hds.com/sites/PMM-sseteam-SIEDev/TW%20Documentation/Puma/Bare%20Metal%20Provisioning/Current%20Draft/Bare%20Metal%20Provisioning%20API%20Reference%20Guide.pdf) **–** Khoa Quach’s document

**Requirements:** Your own copies (2) of Windows Server 2016, which need to be licensed and available for both the

a. Installer content (target computer)--which can be either the officially released Windows Server 2016 Standard Desktop Experience or the Microsoft Windows Server 2016 Standard Desktop Experience evaluation copy, (which is good for 180 days and contains all the features of the official release)-- and

b. [Windows Assessment and Deployment](https://docs.microsoft.com/en-us/windows-hardware/get-started/adk-install) toolkit (reference computer), a Windows 2016 server containing the Windows PE environment used to automate the Windows installer. Although the Microsoft website calls it “ADK for Windows 10” this toolkit is also designed to work with Windows Server 2016. Both operating systems are built on the same foundation.

c. Razor Server Host – CentOS 7.5 was used in this demo

d. One of the following blade types: CB 520H B3, CB 520H B4, CB 520X B2, CB 520X B3

e. Certified Hitachi Compute Blade 500 Drivers

f. Compressed file name ‘win16-razor.tar’ which contains razor associated items listed in table below. Untar to /etc/temp/razor-items

[](https://s3-us-west-2.amazonaws.com/razor-releases/microkernel-008.tar)

g. This step is optional but highly recommended. We want to create a shared directory in Windows so that we can transfer and share files from our Windows 2016 ADK server to our Linux razor-server host. Please refer to the following instructions:

1. On your Windows Server create a shared directory, e.g. c:\temp, and share it with Everyone.
2. Right click on c:\temp and on the screen that says “share with”, do a left click on “Specific People”.
3. In the screen that follows, type in the word, “Everyone” and left click on Share.

Fig. 1

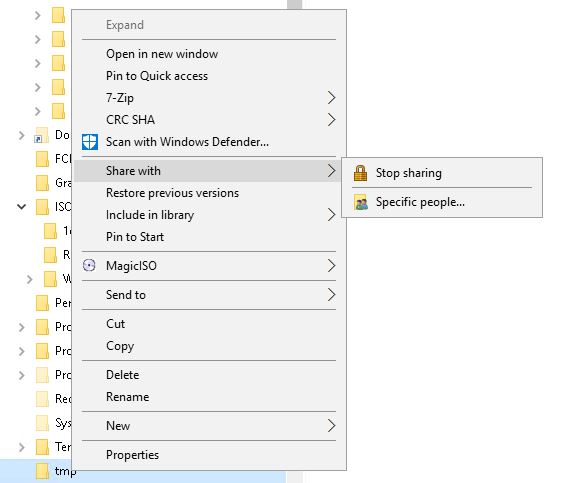
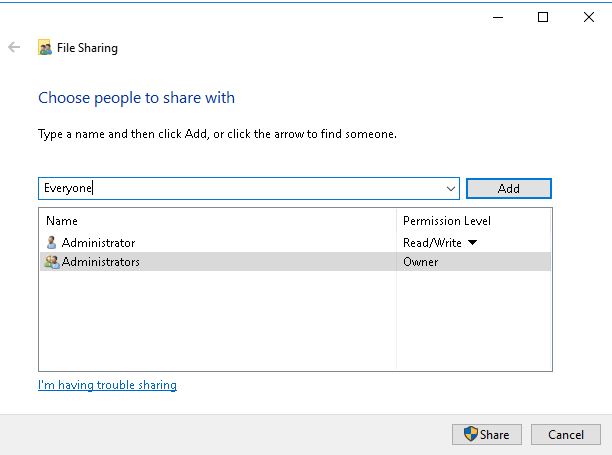


Fig. 2



1. On your razor-server linux host create a directory to share. For example: mkdir /etc/temp
2. Run the following command inserting the correct password and ip address for the razor server:

mount -t cifs -o username=administrator,password=[password] //[ipaddress]/temp /etc/temp

Once the above prerequisites are completed we can move to razor-server installation.

# **Getting Started**

1. Install the razor server and client on your Centos 7.5 server:

yum install <http://yum.puppetlabs.com/puppet5/el/7/x86_64/razor-server-1.8.1.0-1.el7.noarch.rpm> or use the version in the win16-razor.tar file

yum install razor-server

1. Install the razor-client

yum install ruby

yum gem install razor-client

1. Enable, start, and check the status of the razor-server service

systemctl enable razor-server

systemctl start razor-server

systemctl status razor-server

1. Install the Postgres database and initialize the databases

yum install postgresql-server postgresql-contrib

postgresql-setup initdb

1. Configure Postgres to allow remote access by adding the Subnet or IP address of the razor server connecting to Postgres

vi /var/lib/pgsql/data/pg\_hba.conf

host all razor-ip-address/subnet-mask trust

1. vi /var/lib/pgsql/data/postgresql.conf

listen\_addresses = '\*'

1. Enable, start and check the status of postgresql

systemctl enable postgresql

systemctl start postgresql

systemctl status -l postgresql

1. Login to postgres

su – postgres

1. Login to Psql and create razoruser and razor database

Psql

CREATE USER razoruser WITH PASSWORD 'password';

CREATE DATABASE razor\_prd OWNER razoruser;

# to exit postgres cli

\q

1. Configure the razor server database connection string by editing the razor config file

vi /etc/puppetlabs/razor-server/config.yaml

on the line beneath ‘production:’ database\_url, change the password to the actual password

'jdbc:postgresql:razor\_prd?user=razoruser&password=NEWPASSWORD'

do the same on the line that beneath ‘test:’

'jdbc:postgresql:razor\_prd?user=razoruser&password=NEWPASSWORD'

1. Load the razor database schema into PostgreSQL

sudo -u razor razor-admin -e production migrate-database

1. Restart the razor-server and check its status. You may need to run the status -l command several times to see the screen fully populated.

systemctl restart razor-server

systemctl status -l razor-server

1. Install the TFTP Server

Yum install tftp tftp-server xinetd

cd /var/lib/tftpboot

systemctl enable xinetd

systemctl start xinetd

systemctl status -l xinetd

vi /etc/xinetd.d/tftp

change line 14 from disable = yes to disable = no

chgrp -hR razor /var/lib/tftpboot

chown -hR razor /var/lib/tftpboot

1. Generate an iPXE script, and prepare it for TFTP access-

iPXE performs a "chain load" (between the NIC’s regular PXE code and the OS boot code). The script requires the hard coded IP address of the razor server: [change the yellow outline to your own ip address in the following steps]

# env | grep http

# wget http://ip address:8150/api/microkernel/bootstrap?nic\_max=1 [this number represents the number of NIC cards in your blade or server. Maximum number is max=4]

# mv bootstrap\?nic\_max\=1 /var/lib/tftpboot/bootstrap.ipxe

1. Install the microkernel. This will allow the mini-OS in the microkernel to load the repo during deployment of Windows. Use the following command:

tar -xvf  [razor-microkernel-latest --directory=/opt/puppetlabs/server/data/razor-server/repo](https://s3-us-west-2.amazonaws.com/razor-releases/microkernel-008.tar)

or download the microkernel here:

<http://links.puppetlabs.com/razor-microkernel-latest.tar>

unpack it to /opt/puppetlabs/server/data/razor-server/repo

1. Create a broker using the razor client: [this type of broker points to a null value]

razor create-broker --name=noop --broker-type=noop

1. Update /etc/puppetlabs/razor-server/config.yaml and confirm the path entries-the following are typical entries

add the broker path - line 92

broker\_path: /opt/puppetlabs/server/apps/razor-server/share/razor-server/brokers

add the task-path – line 87

task\_path: /opt/puppetlabs/server/apps/razor-server/share/razor-server/tasks

update repo\_store\_root: -- line88

repo\_store\_root: /opt/puppetlabs/server/data/razor-server/repo

update hooks path: -- line95

hook\_path: /opt/puppetlabs/razor-server/hooks

# **Known Issue Requires Attention to complete Razor Setup**

The issue appears to be with these two lines in /opt/puppetlabs/server/apps/razor-server/share/torquebox/jboss/standalone/configuration/standalone.xml:

<socket-binding name='http' port='${http.port:8080}'/>

<socket-binding name='https' port='8443'/>

The port ‘8080’ is incorrectly entered on the first line and the second line ‘https’ is not read at all.

Refer to the following change for the final edit. Check lines 333-347 in /standalone.xml to insure the proper format is used. Note that ports 8080 and 8443 are not required and that the correct format will match the green outline below. Enacting these changes will allow the razor-server service to function correctly.

332 </interfaces>

333 <socket-binding-group name='standard-sockets' default-interface='public' port-offset='${jboss.socket.binding.port-offset:0}'>

334 <socket-binding name='management-native' interface='management' port='${jboss.management.native.port:9999}'/>

335 <socket-binding name='management-http' interface='management' port='${jboss.management.http.port:9990}'/>

336 <socket-binding name='management-https' interface='management' port='${jboss.management.https.port:9443}'/>

337 <socket-binding name='ajp' port='8009'/>

338 <socket-binding name='http' port='8150'/>

339 <socket-binding name='https' port='8151'/>

340 <socket-binding name='messaging' port='5445'/>

341 <socket-binding name='messaging-group' port='0' multicast-address='${jboss.messaging.group.address:231.7.7.7}' multicast-port='${jboss .messaging.group.port:9876}'/>

342 <socket-binding name='messaging-throughput' port='5455'/>

343 <socket-binding name='remoting' port='4447'/>

344 <socket-binding name='txn-recovery-environment' port='4712'/>

345 <socket-binding name='txn-status-manager' port='4713'/>

346 <socket-binding name='stomp' port='8675'/>

347 <socket-binding name='stomp-secure' port='8676'/>

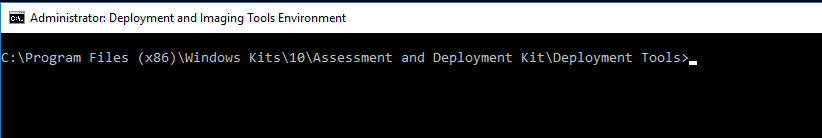
Run systemctl restart razor-server

# **Setting up for Windows installation**

Getting Windows installable is a multi-step process. It is recommended to install the following two items using a Windows Server 2016 OS. Successful installation will provide you with a base with which to build a custom WinPE image suitable for use with Razor:

1. Install the [Windows Assessment and Deployment](https://go.microsoft.com/fwlink/?linkid=2026036) IN THE DEFAULT LOCATION.
2. Now install the [Windows PE-Addon for the ADK](https://go.microsoft.com/fwlink/?linkid=2022233). This set of files is an essential add on for razor integration.
3. Download the windows razor file ‘build-razor-winpe.ps1’ from git hub and copy it to the deployment tools directory on the Windows Server 2016 VM where you have ADK installed:

'C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools'

Open a deployment administrator command prompt. The command prompt is part of the ADK located in folder named below and found in Windows ‘Search’ menu.

1. Create a "dummy file" called razor-client.ps1, do not place any text in it, and save it to 'C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools'
2. Create a directory called 'C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\extra-drivers, subdirectory name is discretionary, and copy the most recently validated Hitachi boot drivers there. This will include all the driver files associated with the validated driver. {show illustrative screenshot]. Run the following command from the ADK command prompt. You will be creating the build script. Do not use the powershell application to run this command or you may not obtain the desired result. Note: It is recommended that you hard code the IP Address or FQDN \* -[NEED TO EDIT and change to NetBios Name] of your razor-server host Centos-VM in the razor-client.ps1 file, during the build of your WinPE image.

powershell -executionpolicy bypass -noninteractive -file build-razor-winpe.ps1 -razorurl http://<RazorServerIPAddress>:8150/svc

1. This command will take some time to complete. It needs to complete without any errors. Output should resemble the following –

**Directory: C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools**

**Mode LastWriteTime Length Name**

**---- ------------- ------ ----**

**d----- 2/6/2019 4:59 PM razor-winpe**

**d----- 2/6/2019 4:59 PM razor-winpe-mount**

**\* Copy the clean ADK WinPE image into our output area.**

**\* Mounting the wim image**

**Path : C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\razor-winpe-mount**

**Online : False**

**\* Adding powershell, and dependencies, to the image**

**\*\* Installing WinPE-WMI to image**

**Path : C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\razor-winpe-mount**

**Online : False**

**\*\* Installing WinPE-NetFX to image**

**Path : C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\razor-winpe-mount**

**Online : False**

**\*\* Installing WinPE-Scripting to image**

**Dismount-WindowsImage**

**Running**

**[oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo ]**

**Path : C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\razor-winpe-mount**

**Online : False**

**\*\* Installing WinPE-PowerShell to image**

**Driver : oem0.inf**

**OriginalFileName : C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment**

**Tools\razor-winpe-mount\Windows\System32\DriverStore\FileRepository\hfcwdd.inf\_amd64\_c4a0c9c58b6134b7\hfcwdd.inf**

**InBox : False**

**CatalogFile : hfcwdd.cat**

**ClassName : SCSIAdapter**

**ClassGuid : {4D36E97B-E325-11CE-BFC1-08002BE10318}**

**ClassDescription : Storage controllers**

**BootCritical : True**

**DriverSignature : Signed**

**ProviderName : Hitachi Ltd.**

**Date : 8/24/2016 12:00:00 AM**

**Version : 4.5.8.2240**

**\* Writing startup PowerShell script**

**\* Writing Windows\System32\startnet.cmd script**

**\* Unmounting and saving the wim image**

1. When the razor build script completes, it will produce a new directory called \razor-winpe\ under the current working directory, C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\**,** and will contain a file named boot.wim**.** This is your custom WinPE image containing the required components used to work with the Razor server. It is the razor equivalent of \sources\boot.wim that is packaged inside the Windows 2016 Server installation DVD and is a vital part of the automation process. If you need to run the command again make sure the \razor-winpe\ directory does not exist before executing it.
2. Next, copy C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\razor-winpe\boot.wim to C:\WinPE\_amd64\media\sources\boot.wim and either save or overwrite the original.
3. Next, copy build-razor-winpe.ps1 and razor-client.ps1 to C:\WinPE\_amd64
4. \* Optional Step:

Create a bootable iso file using the following command at a Deployment Tools command prompt:



1. \* Optional Step:

The ISO file will be created in C:\WinPE\_amd64\ directory. Test the iso by attaching it to an available Hitachi blade virtual drive and insure that it boots your blade correctly. If necessary, reinstall the CB500 drivers or obtain the ones designed for your blade and test again. Your blade boot order needs to be set to boot CD ROM first.

1. \* Optional Step:

Wait for the Microsoft WinPE command prompt to appear. You will now prepare to collect the information to create the startnet.cmd script to help kickstart your Windows network from within razor.

**Create the Startnet.cmd Script - \* note:** this script can be created within the build-razor-winpe.ps1 Use the startnet.cmd commands and go to lines: [enter the line numbers here to show how add the script]

The startnet.cmd script is a kickstart file that allows you to connect to your network and begin deploying Windows Server 2016 after a WinPE boot up. This script needs to be added to the boot.wim file using the DISM tool included in the ADK. Follow the steps below:

1. Run this command:

netsh interface ipv4>show addresses [this command will get the names of the LAN cards to which we will attach an IP address].

1. The command will produce output similar to the following on each of the interfaces. For convenience, only the output from the first interface is listed.

Configuration for interface "Onboard LAN 0-0 Func 0"

DHCP enabled: Yes

IP Address: 169.254.145.149

Subnet Prefix: 169.254.0.0/16 (mask 255.255.0.0)

InterfaceMetric: 15

1. Using interface id for the first NIC – "Onboard LAN 0-0 Func 0" assign a static IP address (the following ip’s are examples only. Use actual IP addresses from your current network)

netsh interface ipv4 set address "Onboard LAN 0-0 Func 0" static 10.17.10.17 (ip address to assign) 255.255.255.0 (subnet mask) 10.17.10.1 (gateway ip address)

1. After the IP is assigned, ping the gateway to verify that you have network connectivity.

ping 10.17.10.1

1. The information gleaned will be used to complete the startnet.cmd script that will automate the Windows installation in Razor.

**How do I utilize startnet.cmd? Where is it located?**

Startnet.cmd is located in the \windows\system32 folder of the boot.wim image file, a compressed file that is located in the \media\sources directory of the WinPE.iso

In order to edit the startnet.cmd the boot.wim image is mounted using the DISM tool located in the ADK. The startnet.cmd needs to be customized to your environment. Below is an example of the commands used to assign an IP address to the first NIC in the blade. Please use ip addresses that suit your network requirements. The explanations after each command line are not part of the script.

-------------------------

wpeutil [starts the wpeutility]

netsh interface ipv4 set address "Onboard LAN 0-0 Func 0" static 10.17.10.17 255.255.255.0 10.17.10.1 [assigns the IP address to LAN 0-0]

ping 127.0.0.1 -n 16 -w 1000>null [tests connectivity in the network using a timeout designed to allow ‘wait time’ until the interface comes up]

net use p: \\10.17.10.116\temp /User:Administrator Hitachi1 [connects to a share on the network that contains the Windows Server 2016 installation files]

---------------------------

Example of actual script:

@echo off

echo starting wpeinit to detect and boot network hardware

wpeinit

netsh interface ipv4 set address "Onboard LAN 0-0 Func 0" static 172.17.11.117 255.255.255.192 172.17.11.1

ping 127.0.0.1 -n 16 -w 1000>null

net use p: \\172.17.11.116\temp /User:Administrator Hitachi1

echo starting the razor client

powershell -executionpolicy bypass -noninteractive -file %SYSTEMDRIVE%\razor-client.ps1

echo dropping to a command shell now...

p:/Win2016/setup.exe

In the next section we will return to the razor server so that we may prepare it to install Windows.

**Create a Repo to use for installing Windows**

Razor stores the contents of OS iso files in the repo folder. In razor documentation, the root directory for the repo files is known as the ‘repo store root’. This is ***not*** the actual name.

The repo directory resides in the following path: /opt/puppetlabs/server/data/razor-server/repo

1. If it is not already there, create the repo directory using the following commands:

cd opt/puppetlabs/server/data/razor-server/

mkdir repo

1. Next, run the ‘razor create-repo’ command using the ‘ --no-content option’

The command will create a shell directory where you will manually unpack the iso files or create and use a script that will mount the WinPE iso and then unpack it to the repo location

Example command:

razor create-repo --name WinPE2016 --no-content --task windows/2016

Successful output from the razor-server will be:

From http://localhost:8150/api/collections/repos/WinPE2016:

name: WinPE2016

iso\_url: ---

url: ---

task: windows/2016

command: http://localhost:8150/api/collections/commands/108

1. Copy the file WinPE\_Razor\_amd64.iso from step 11 of “**Setting up for Windows Installation**” in this doc, to your /etc/tmp shared folder.
2. On your Windows server, download a copy of the Windows 2016 Data Center Evaluation Server distribution iso and place it in a new folder within your Windows share directory. For example, c:\tmp\Win16-Dist
3. Next, mount the iso file to facilitate copying to the repo directory.
4. Create the mount point directory, where you will mount the image:

mkdir /mnt/

1. Create subdirectory /iso/

cd mnt

mkdir iso

1. Mount the file and check its contents:

mount -o loop /etc/tmp/WinPE\_amd64.iso /mnt/iso

1. Go inside the mounted directory (cd/mnt/) and copy the files to your /repo/ directory:

cp -pRf /mnt/iso/\* /opt/puppetlabs/server/data/razor-server/repo/’repo-name’

1. Now unmount the iso: $umount /mnt/
2. Copy the Boot.wim you created in Step 9 of “**Setting up for Windows Installation**” in this doc, to the root of your repo:

cp /tmp/boot.wim /opt/puppetlabs/server/data/razorserver/repo/’repo-name’/boot.wim

### Create SMB Share

Since the WinPE environment can't use an HTTP source for installation we need to create and configure an SMB server on the Razor server, and then export the Razor repositories.

On the share, we want to allow anonymous access, call the share ‘razor’ and point it to your repo store root directory. Default is: smb.conf (e.g. /etc/samba/smb.conf) should look like this:

[razor]

comment = Windows Installers

# path is, by default, under /opt/puppetlabs/server/data/razor-server

Path = /opt/puppetlabs/server/data/razor-server/repo

guest ok = yes

writable = no

browsable = yes

[global]

security = user

#Use defaults for the remainder of values

### The following command will indicate that the share was created:

smbclient -L Lee-Centos7 -I 172.17.11.119 -t 45

Enter SAMBA\root's password:

Anonymous login successful

Sharename Type Comment

--------- ---- -------

print$ Disk Printer Drivers

razor Disk Windows Installers

IPC$ IPC Service (Samba 4.8.3)

Reconnecting with SMB1 for workgroup listing.

Anonymous login successful

### Install and Configure a DHCP Server (dnsmasq)

Razor PXE boot will require a DHCP Server. Our model utilizes dnsmasq.

1. Use YUM to install dnsmasq: yum install dnsmasq
2. If it does not already exist create the directory var/lib/tftpboot:

mkdir /var/lib/tftpboot

1. Change the permissions for /var/lib/tftpboot: chmod 655 /var/lib/tftpboot
2. Add the following items to /etc/dnsmasq.d/dnsmasq.conf

bind-interfaces

# start ip address, end ip address, examples and lease time

dhcp-range=10.10.10.5,10.10.10.10,1h

dhcp-authoritative

dhcp-match=set:UEFI,option:client-arch,7

dhcp-match=set:UEFI,option:client-arch,9

dhcp-match=set:iPXE,175

dhcp-boot=tag:UEFI,snponly.efi

dhcp-boot=tag:iPXE,bootstrap.ipxe

dhcp-boot=undionly.kpxe

enable-tftp

tftp-root=/var/lib/tftpboot

log-facility=/var/log/dnsmasq.log

#Disable dnsmasq's DNS service.

port=0

dhcp-option=option:dns-server,10.105.5.254,10.105.25.254 #example ip addresses

dhcp-option=3,10.5.5.5 #the ‘3’, preceding the example ip address is required

1. Create two additional files in the dnsmasq.d folder. Call one dnsmasq-allows.conf and the next one dnsmasq-ignores.conf

The files can be used to add and register MAC addresses with dnsmasq to help with the network boot process.

Here is an example entry:

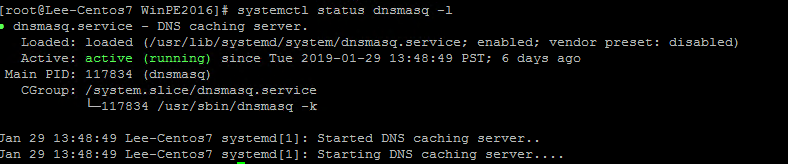
dhcp-host=f8:48:97:7c:ac:40,id:\*,set:test\_provisioning

dhcp-host=d8:c4:97:19:c1:a3,id:\*,set:test\_provisioning

1. Enable, start and check the status of the dnsmasq service with these commands:

systemctl enable dnmasq

systemctl start dnsmasq

systemctl status dnsmasq -l 

1. Refer to <https://wiki.archlinux.org/index.php/dnsmasq#Dynamic> for additional information.

### Check to see if razor tasks were created

Run the command-

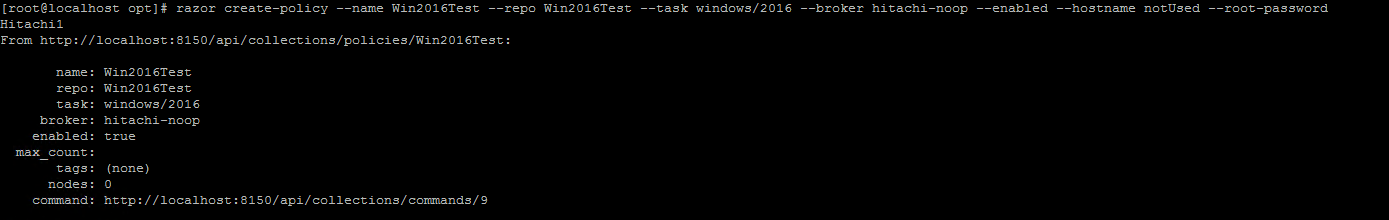
[root@localhost opt]# razor tasks

[it will produce the following screen. You will need the task name to create the policy in the next step]

### 

### Create razor policies

Finally, create your policies as per following example:



The broker will be installed and run as a SetupComplete.cmd script when the install is complete. Unless you are passing the system off to a config management system, creating a "noop" broker will suffice (razor create-broker --name noop --broker-type noop).

# **Using the Windows installation in Razor**

Once you have a policy created for the Windows installation, insure that you have attached the correct repo, tags and tasks.

Run the command ‘razor nodes’ to identify any existing nodes.

If the node you see is attached to a current policy, run the command

‘razor reinstall-node [node#]’ to clear the flag.

Reboot your blade, and insure that PXE boot is selected as the first choice in EFI settings on the blade. Your blade will run the general razor installation OS which will then locate your repo and bind the Windows policy to the node. You should watch the console –and ‘snip’ screenshots to capture errors -- while testing it out.

Log messages will appear in /var/logs/messages on your razor server

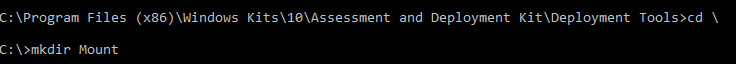
**How do I make changes to the razor-winpe\boot.wim in order to edit the startnet.cmd script?**

Use the dism tool that comes with the ADK. It can be accessed through the Deployment and Imaging Tools Environment command prompt. Dism is used to mount the wim files and after they are mounted, you can edit the startnet.cmd script and/or insert driver files as required.

Dism has a plethora of options and switches to help manipulate image files.

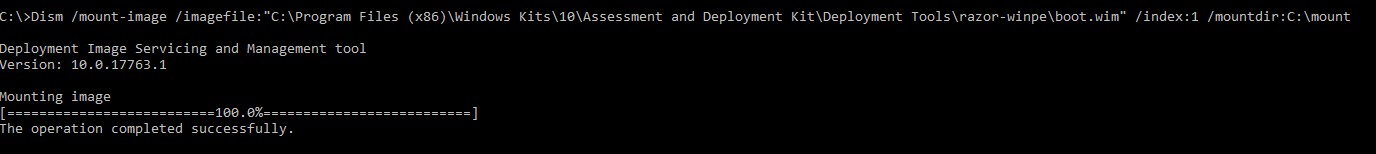
1. Create a mount directory on the C: drive – at a Deployment Tools Command Prompt type:

cd \

Mkdir Mount

1. Mount the boot.wim that was created in the \razor-winpe folder using the following command within the Deployment Tools prompt:

C:\>Dism /mount-image /imagefile:"C:\Program Files (x86)\Windows Kits\10\Assessment and Deployment Kit\Deployment Tools\razor-winpe\boot.wim" /index:1 /mountdir:C:\mount



1. You are now able to edit the startnet.cmd file. From the command prompt, again, type: C:\>notepad c:\mount\windows\system32\startnet.cmd

Edit the script per the instructions listed in the previous section. Save and close the file.

1. Unmount the boot.wim using this command: (make sure to include the /commit flag)

dism /unmount-image /mountdir:c:\mount /commit

**How to reate new key on razor**

openssl genrsa -out newKey.pem

openssl req -new -subj /CN=172.17.11.116/ -key newKey.pem -x509 -days 730 -out newCert.pem

/usr/bin/puma\_adm --configure --key /etc/temp/newKey.pem --cert /etc/temp/newCert.pem --ipaddress 172.17.11.115