

White Paper:
Getting Started with
VMware's PowerShell Toolkit (BETA)

Document Version 1.0

RTFM Education

Beyond the Manual... with Mike Laverick

By Mike Laverick
© RTFM Education

Contact:

mikelaverick@rtfm-ed.co.uk

RTFM Education
www.rtfm-ed.co.uk

ESX Version:

3.x

Author:

Mike Laverick 🛠️

Audience:

Experienced

Style:

Bit techy with some humour, not a po-faced article. I use unhappy faces ☹️ to flag up events and experiences that are less than pleasant. I use a happy face 😊 to flag up something which is very advantageous.

Objective:

To get you up and running with VMware's new PowerShell Toolkit – without having to learn any powershell code!

I was initially put off using the various SDKs, Perl Toolkits and PowerShell Toolkits – although I'm pretty good with the CLI of an ESX host and write useful "shell" (.sh) scripts – I wouldn't say I was a VBS guru or C# guru. Then I discovered PowerGUI. PowerGUI puts a graphical shell around Microsoft PowerShell. It's a bit like the old style "macro" recorders you used to get in spreadsheet or a word processor. You click about making changes – and this creates the code you need. Sure if you want to really fancy things like loops and error control you then need higher level knowledge – but this really great for poor little admins like me who's eyes glaze over once some talks about object, properties, attributes and methods

Installing PowerShell Tools

Downloading the Tools

The tools you need to get up and running are not available as single download from one web-page – but are actually distributed across a number of vendors and enthusiasts web pages. They also need to be installed in the correct order. This is what you need to get started

1. Microsoft PowerShell V1.0. This is downloadable from MS website. You don't need this if you are running Windows 2008. There is a client version (XP/Vista) and a server edition for Windows 2003

<http://www.microsoft.com/windowsserver2003/technologies/management/powershell/download.msp>

2. The next thing you need download is VMware PowerShell which is currently in Beta. This extends Microsoft PowerShell to be "VMware Aware". It allows you to script the high-level management of VirtualCenter and Virtual Machine

<http://www.vmware.com/support/developer/windowstoolkit/beta/windowstoolkit-200803-releasenotes.html>

3. The next component you will need is some software from Quest called ActiveRoles Management Shell for Active Directory. These extensions make authenticating via Active Directory – and handling Active Directory issues easier. You can download it from here:

<http://www.quest.com/powershell/activeroles-server.aspx>

4. The next component that is really, really useful is PowerGUI. It's this adds a GUI wrapper to PowerShell so you don't have to learn or manually type all that nasty code.

<http://www.powergui.org/downloads.jspa>

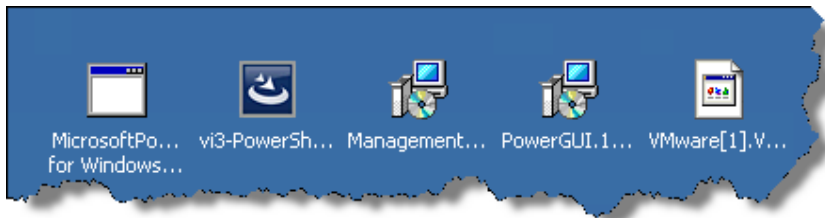
5. And Finally!!! The next thing you will need is the VMware "PowerPack". This hasn't been written by VMware, but by enthusiast called Dmitry Sotnikov. It adds additional "node" to the PowerGUI utility called "VMware" and contains lots of tasks that you want to automate. This shields you from having to learn the VMware PowerShell from scratch

<http://powergui.org/entry!default.jspa?categoryID=21&externalID=1802>

Warning:

Right now the PowerPack for VMware doesn't have every single attribute that you could change. Heck, these Dmitry has done it for free. I imagine he is working on, testing a new PowerPack which will have more options. You might be interested to know there's a PowerPack for VMware Lab Manager Administration by Mark Kilfoil. You can find them both in the Virtualization category on the PowerGUI website

Once you have downloaded all these packages and zip files we can begin installation.



Installing the Tools

For the most part each of these tools can be installed in a next-next – yeah-yeah – style. Occasionally, though there will be some post configuration tasks, before moving on to the next stage.

The install of the Microsoft PowerShell environment is one those next-next-yeah-yeah installs. It must be installed BEFORE the VMware PowerShell software. Otherwise the VMware installer will just barf up anyway.

1. Install the Microsoft PowerShell software

Note:

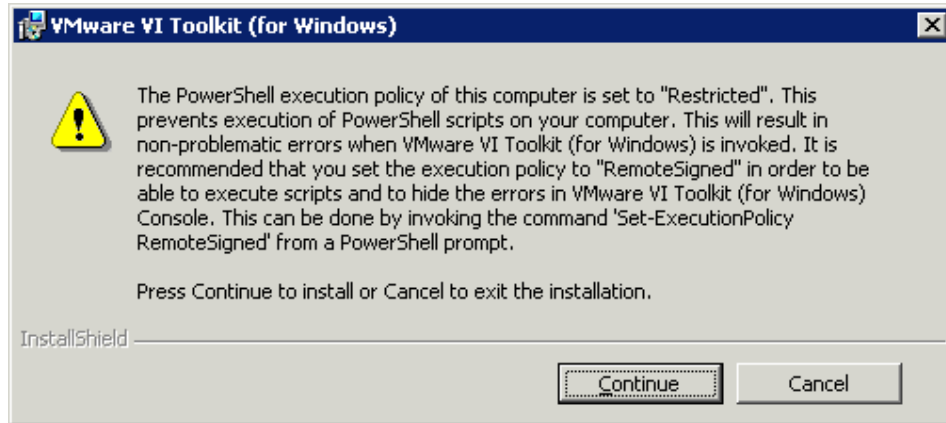
Curiously MS PowerShell's installer is like a service pack – it also extracts some read me files which are probably worth a look at some time. I didn't bother.

The next thing we install is VMware's PowerShell. Although it's another next-next-yeah-yeah install there is a little bit of post-configuration. PowerShell appears to use certificate signing on software, and VMware use one of their "auto-generated" certificates to this. This VMware Certificate must be trusted before its run. Additionally, we have to adjust the security settings - by default security settings are set to "restricted" which limits the functionality of the PowerShell libraries to the system they were installed to. We can lower this security restriction to stop unwanted errors.

2. Install VMware PowerShell software

Note:

As mentioned in the paragraph above the installer does warn you about security defaults which restrict functionality

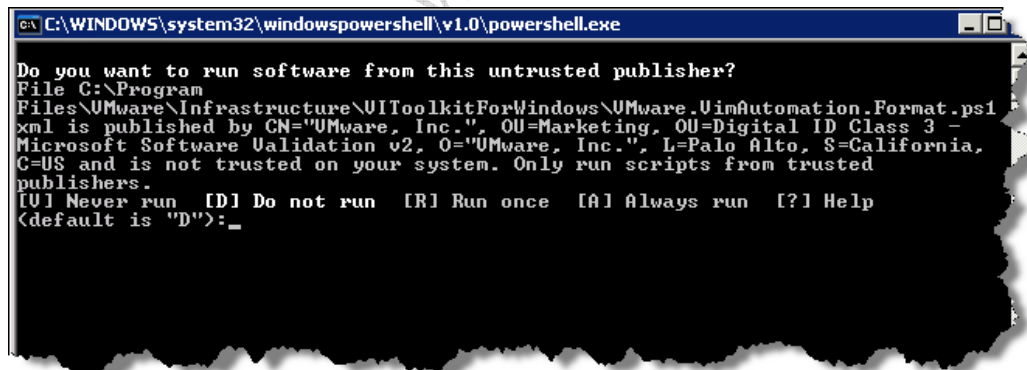


3. Then next-next-yeah-yeah your way through the remaining questions. This will leave you with a shortcut on your desktop to open a PowerShell Session like so:



Note:

When you run the PowerShell for the first time – you will receive a certificate prompt



Choose [A] to allow VMware's PowerShell libraries to function – trusting the auto-generated certificate and stopping this warning from appearing again. The first time you do this – you will get an error

```
C:\WINDOWS\system32\windowspowershell\v1.0\powershell.exe
Do you want to run software from this untrusted publisher?
File C:\Program
Files\VMware\Infrastructure\UIToolkitForWindows\VMware.VimAutomation.Format.ps1
xml is published by CN="VMware, Inc.", OU=Marketing, OU=Digital ID Class 3 -
Microsoft Software Validation v2, O="VMware, Inc.", L=Palo Alto, S=California,
C=US and is not trusted on your system. Only run scripts from trusted
publishers.
[I] Never run [D] Do not run [R] Run once [A] Always run [?] Help
(default is "D"):a
File C:\Program Files\VMware\Infrastructure\UIToolkitForWindows\init.ps1 cannot
be loaded because the execution of scripts is disabled on this system. Please
see "get-help about_signing" for more details.
At line:1 char:2
+ . <<<< "C:\Program Files\VMware\Infrastructure\UIToolkitForWindows\init.ps1"
PS C:\WINDOWS> _
```

Let input the variable mentioned in the installer to stop this from happening. Type

Set-ExecutionPolicy RemoteSigned

Warning:

In the warning dialog box that VMware has in their install – the dialog box doesn't show the space in this command. There is a space between Set-ExecutionPolicy and policy setting of RemoteSigned

Note:

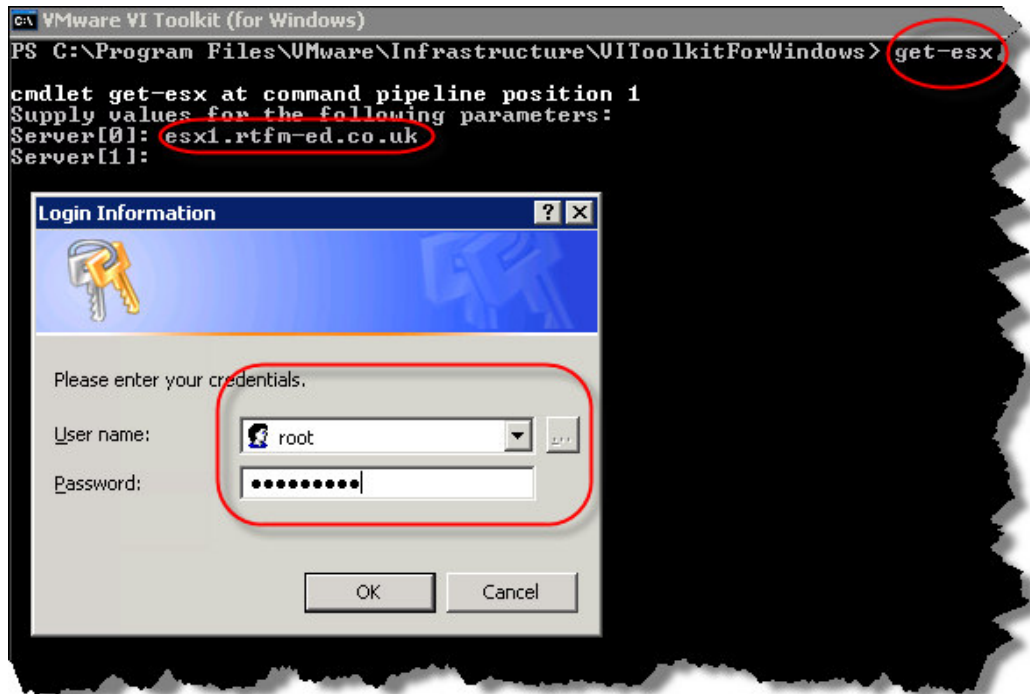
After doing this the VMware PowerShell window should open without warnings or annoying red error messages

```
VMware VI Toolkit (for Windows)
Useful starting points:
Get-UICommand - returns a list of available VMware Infrastructure commands
Get-UC or Get-ESX - logs in to a UC or ESX server
Get-UM - finds available virtual machines
PS C:\Program Files\VMware\Infrastructure\UIToolkitForWindows> _
```

TIP:

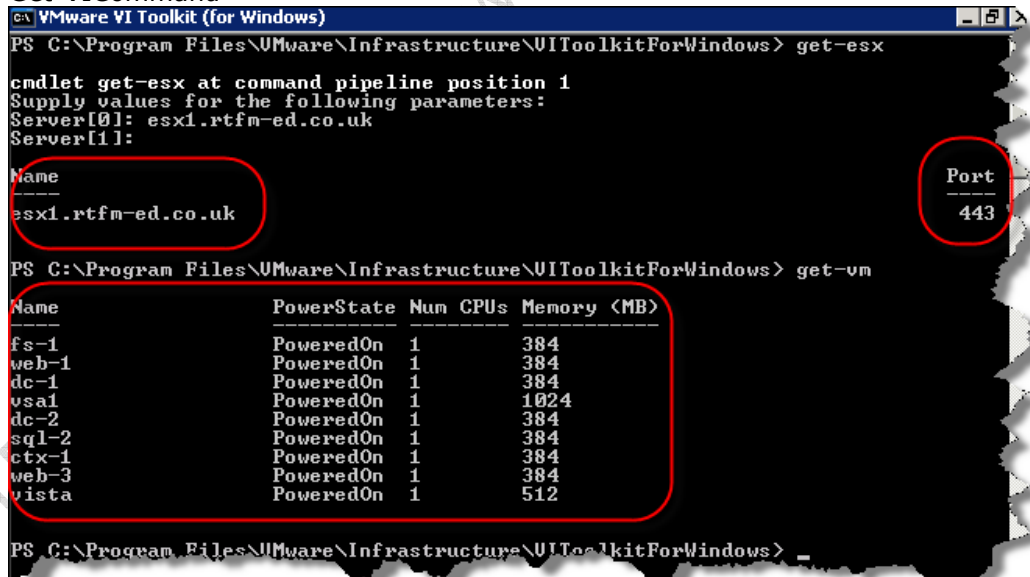
If you want to quickly test if the VMware PowerShell is working. Do the following

```
get-esx
provide the login credentials to the ESX
get-vm
```



TIP:

For a complete list of all the cmdlets from the VMware Powershell, type `Get-VICommand`



- Next **Install the ActiveRoles Management Shell for Active Directory** software from Quest. This another next-next-yeah-yeah installer that requires for no special post-configuration
- Install the PowerGUI** Software

Note:

The PowerGUI installer does allow to add in extensions (PowerPacks) for

Exchange and Operations Manager if you want them

6. **Launch the PowerGUI Application** from your start menu the
7. Finally, **right click the + PowerGUI node** from the application and choose **Import**

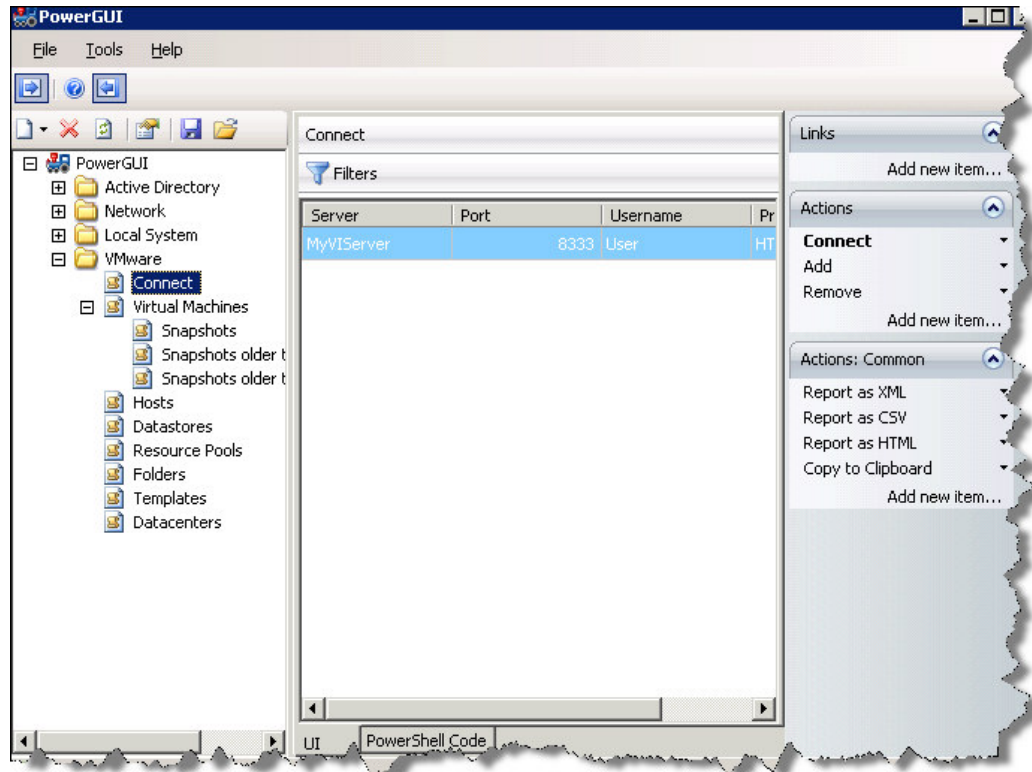


8. **Browse to find the VMware[1].VIToolkit.powerpack file** and click **Open**

Note:

This should add a +node to the PowerGUI application called +VMware like so:

RTFM Education - VMware



Using PowerShell Tools

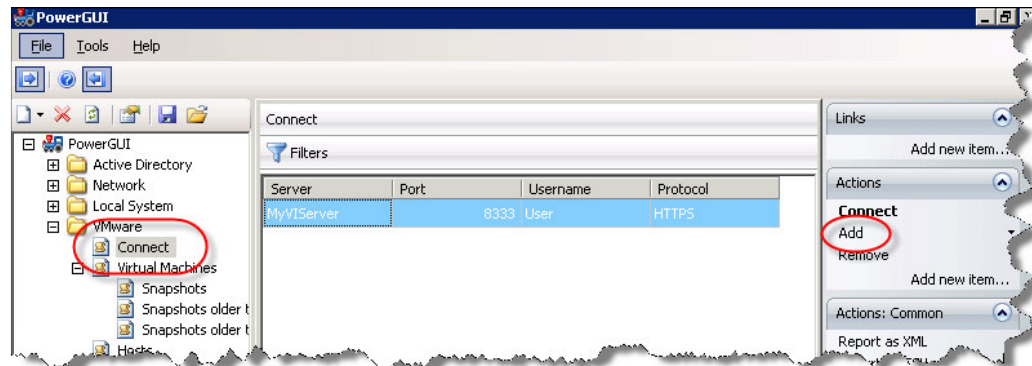
I'm still myself getting to grips with PowerShell and PowerGUI. But I just want to walk you through using the application with a couple of neat examples. What got me started looking at this issue was that I had whole series of virtual machines that were patched into portgroup called vlan21, and I wanted to patch them into a new VLAN called vlan61. Doing this task by hand was going to take a long time, and found simply editing the vmx file did not produce the result I was looking for. So I began asking people on the forums of ways to do this – and PowerShell method came to my attention. What attracted it to me PowerGUI and the various extensions and powerpacks we have installed was how easy it was.

Authenticating to the VirtualCenter

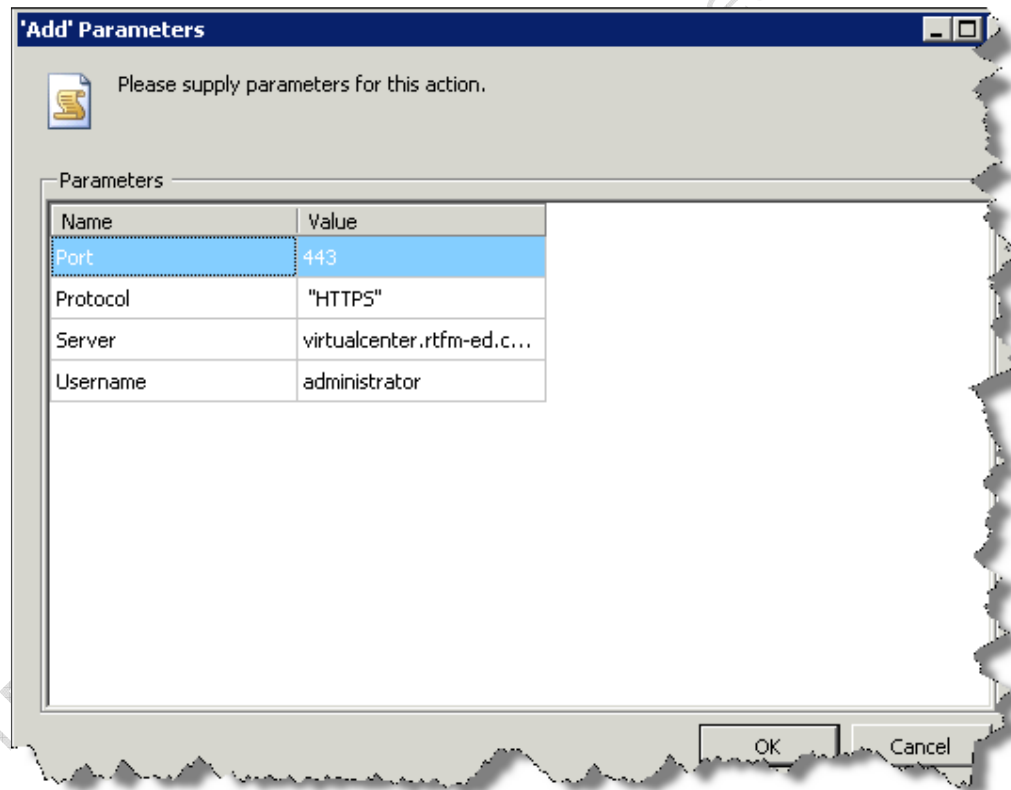
One of the first things you have to do is connect and authenticate either to an ESX host or a VirtualCenter server. Fortunately, PowerGUI has the built in to the UI and create PowerShell code for you.

1. In **PowerGUI**, Select **+VMware** and select **Connect**

2. Over on the right-hand side select the **Add Button**

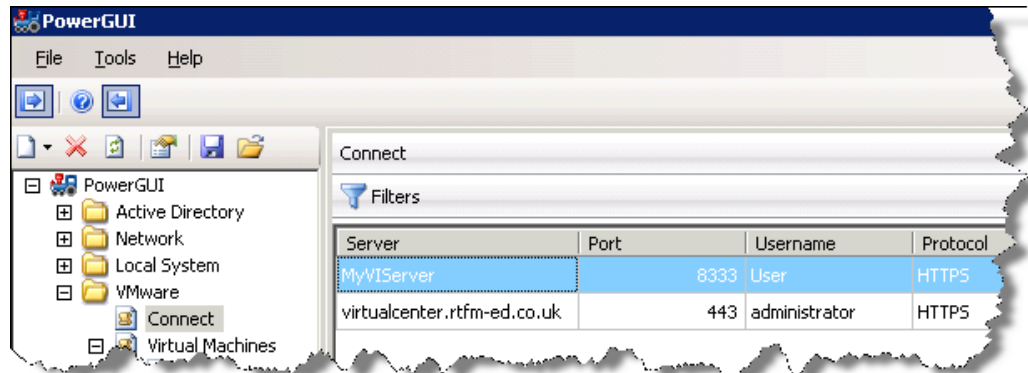


3. In the **Server field**, type in the **FQDN or IP address** of your **ESX host/VirtualCenter**
4. In the **Username field**, type in the **user account to authenticate to the system** and Click **OK**



Note:

This entry should then appear in the list of connections



5. If you **double-click your entry**, you should be **challenged for a username and password to connect** to the VirtualCenter/ESX system



Note:

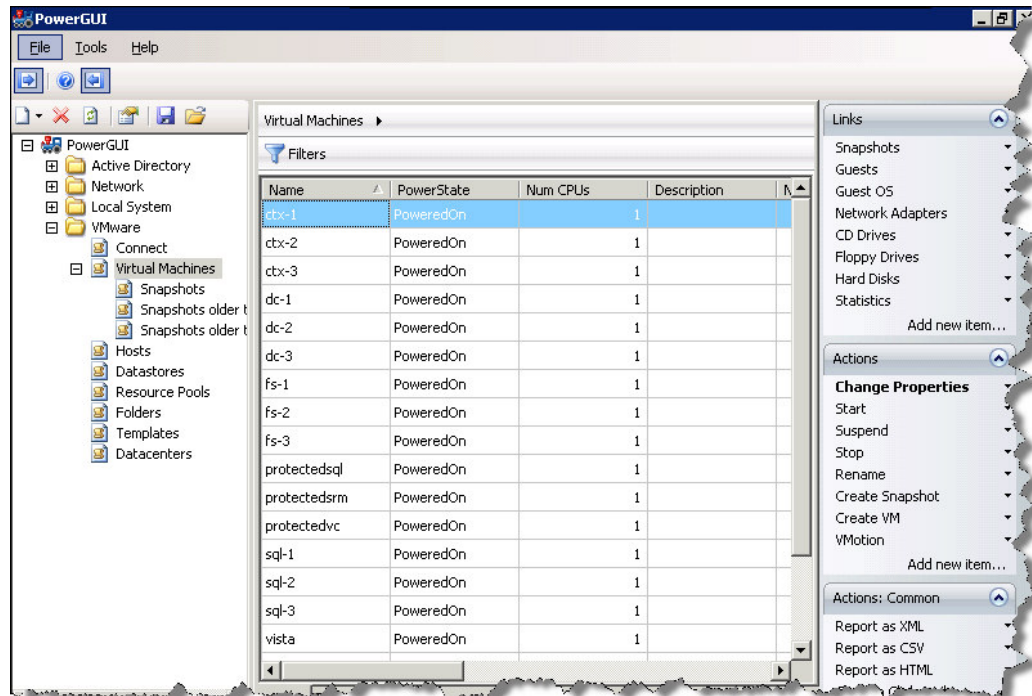
After completing the login, you should get a progress bar in the bottom right-hand corner as it connects to ESX/VirtualCenter



Change the Port Group Assignment for Multiple Virtual Machines

After you have connected to the ESX/VirtualCenter system – you should see your virtual machines in the PowerGUI Application

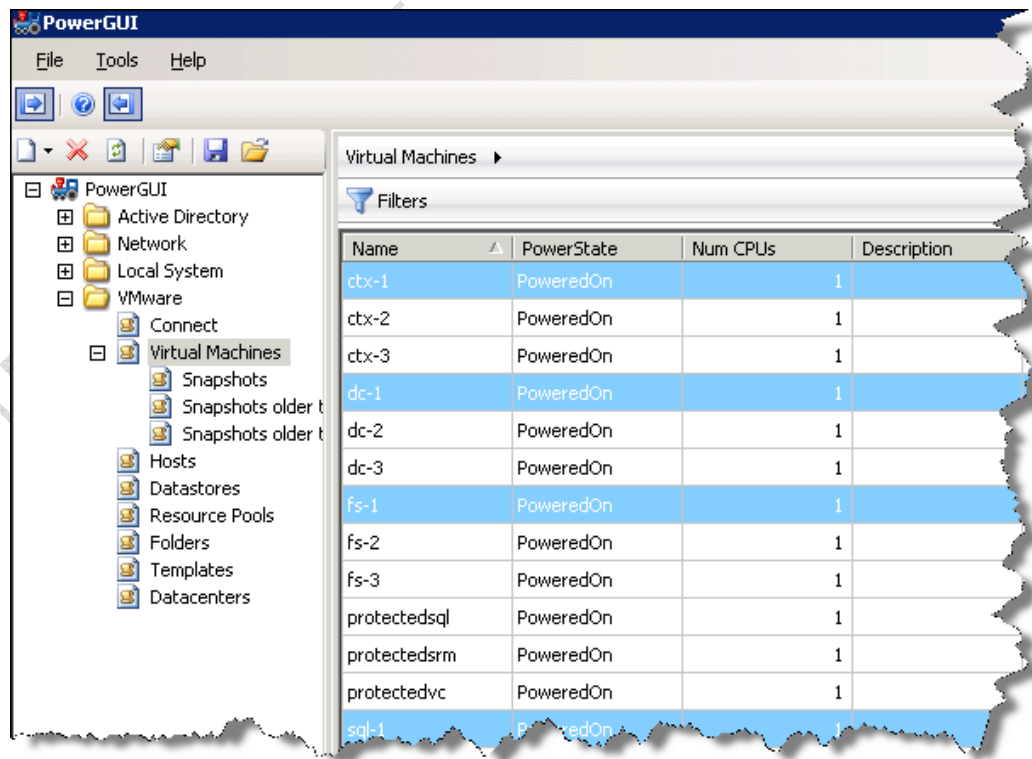
1. **In the PowerGUI application**, select the **+Virtualmachines node**. This should refresh the middle pane showing the virtualmachines



Note:

As you can see there is a huge amount of Links and Actions we can use.

2. In the **PowerGUI** interface it is possible to do multiple selections with **[ctrl]+click**

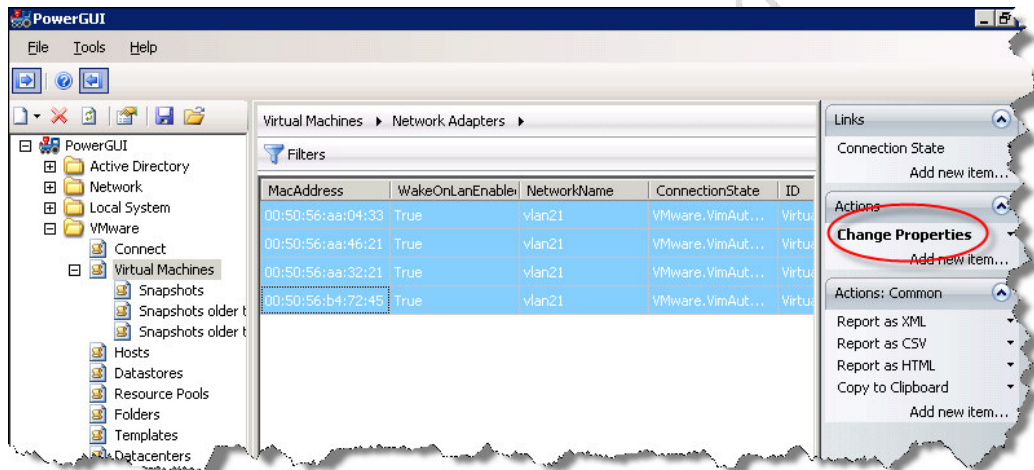


3. **Once you have selected more than one virtual machine**, choose the **Network Adapters** option on the right hand side

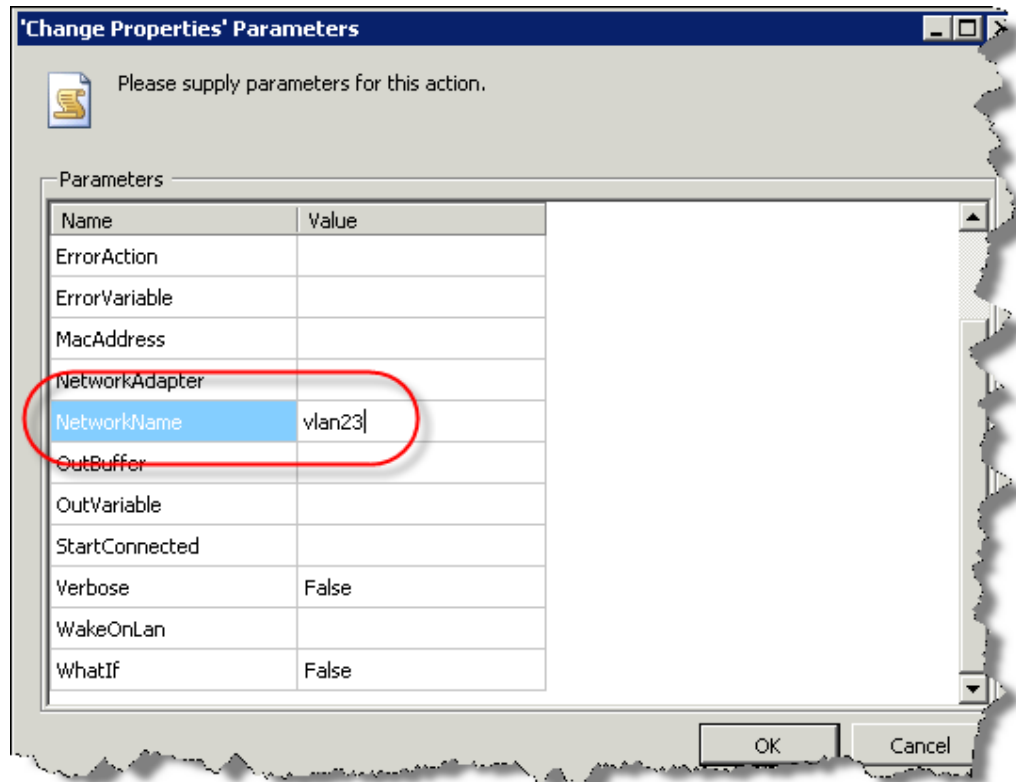


This should open another dialog box that shows the network settings of the selected virtual machines. Notice how you can see the VLAN port group settings in the NetworkName location

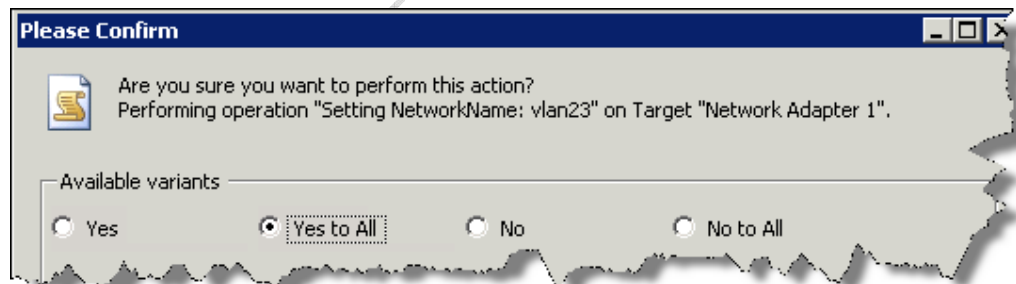
4. Next select the **Change Properties** option



5. **Scroll down the dialog box** and located the **"NetworkName"** attribute, and **type in the new port group name**



6. In the **confirmation dialog box**, choose **Yes to All**, to allow this modification to effect all selected virtual machines



Note:

Before you click OK. You might like to have the Vi Client open before you click OK, so you can watch the activity that's about to take place. What you will see is that VirtualCenter receives the PowerShell instructions and updates the configuration of all the virtual machines like so

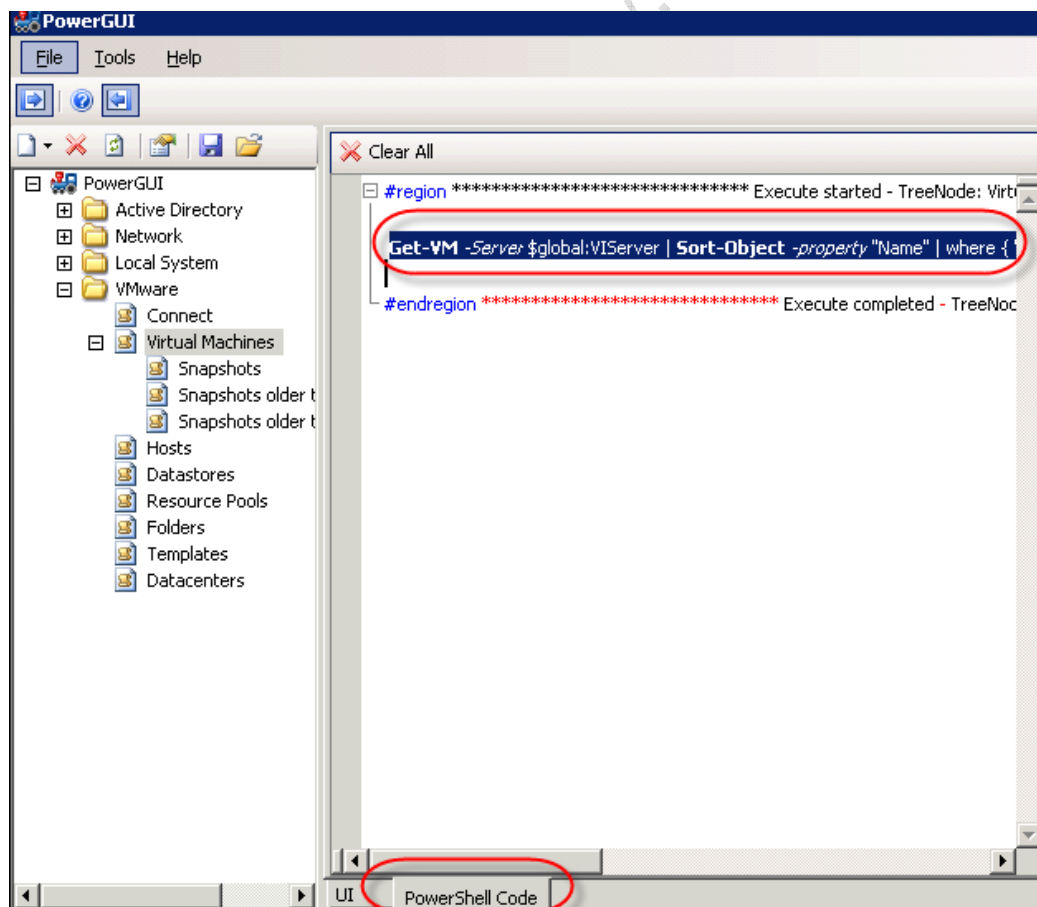
Recent Tasks		
Name	Target	Status
Auto-Modify Virtual Ma...	web-1	Completed
Auto-Modify Virtual Ma...	sql-1	Completed
Auto-Modify Virtual Ma...	dc-1	Completed
Auto-Modify Virtual Ma...	ctx-1	Completed
Reconfigure Virtual Ma...	web-1	Completed
Reconfigure Virtual Ma...	sql-1	Completed
Reconfigure Virtual Ma...	dc-1	Completed
Reconfigure Virtual Ma...	ctx-1	Completed

Seeing the PowerShell Code

The actual code required to do this is in fact one line of PowerShell script

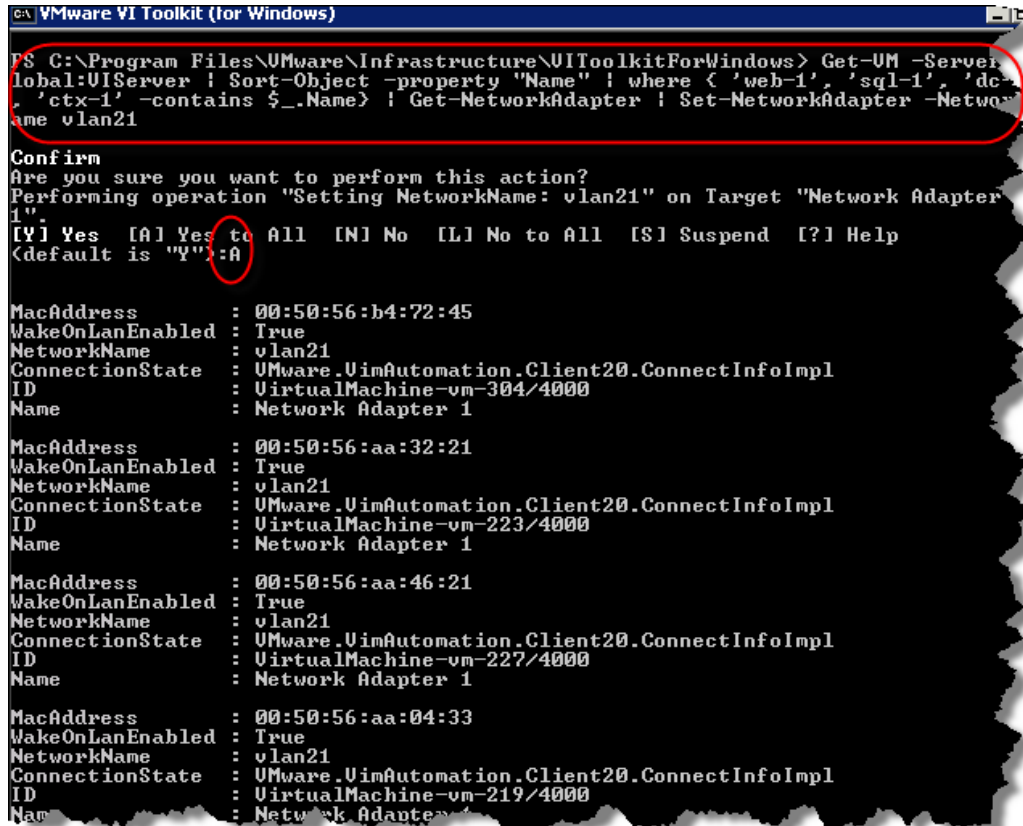
```
Get-VM -Server $global:VIServer | Sort-Object -property "Name" | where { 'web-1', 'sql-1', 'dc-1', 'ctx-1' -contains $_.Name } | Get-NetworkAdapter | Set-NetworkAdapter -NetworkName vlan23
```

You can see the PowerShell code from the "PowerShell" tab which sits next to the UI tab at the bottom of the PowerGUI Application



If you have clicked about a lot in PowerGUI session, and tried didn't changes many times – you will have a lot more code. It's not unlike a macro recorder. Every action in the GUI – good AND bad is "recorded" in the PowerShell Code tab

This exact same "code snippet" could be used directory at a PowerShell command-prompt, as the screen dump below shows



```
PS C:\Program Files\VMware\Infrastructure\UIToolkitForWindows> Get-UM -Server
Global:UIServer | Sort-Object -property "Name" | where { 'web-1', 'sql-1', 'dc-1', 'ctx-1' -contains $_.Name } | Get-NetworkAdapter | Set-NetworkAdapter -NetworkName vlan21
```

Confirm
Are you sure you want to perform this action?
Performing operation "Setting NetworkName: vlan21" on Target "Network Adapter 1".
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help
(default is "Y"):A

MacAddress : 00:50:56:b4:72:45
WakeOnLanEnabled : True
NetworkName : vlan21
ConnectionState : VMware.UimAutomation.Client20.ConnectInfoImpl
ID : VirtualMachine-vm-304/4000
Name : Network Adapter 1

MacAddress : 00:50:56:aa:32:21
WakeOnLanEnabled : True
NetworkName : vlan21
ConnectionState : VMware.UimAutomation.Client20.ConnectInfoImpl
ID : VirtualMachine-vm-223/4000
Name : Network Adapter 1

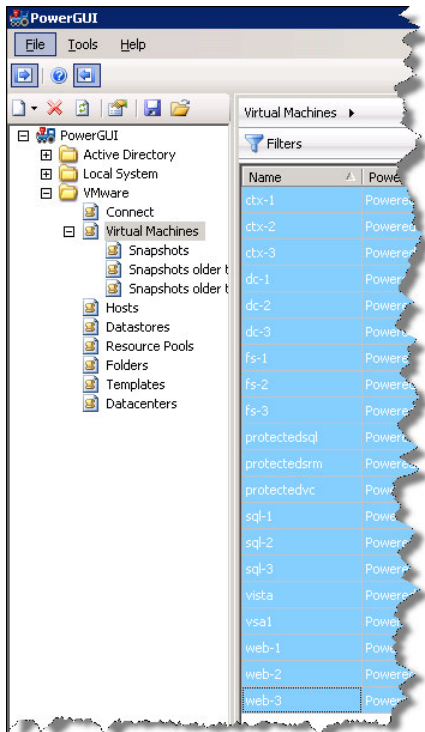
MacAddress : 00:50:56:aa:46:21
WakeOnLanEnabled : True
NetworkName : vlan21
ConnectionState : VMware.UimAutomation.Client20.ConnectInfoImpl
ID : VirtualMachine-vm-227/4000
Name : Network Adapter 1

MacAddress : 00:50:56:aa:04:33
WakeOnLanEnabled : True
NetworkName : vlan21
ConnectionState : VMware.UimAutomation.Client20.ConnectInfoImpl
ID : VirtualMachine-vm-219/4000
Name : Network Adapter 1

Advanced Search and Selection

In the above example the tricky thing is having to select the virtual machines. There is a better where of doing this.

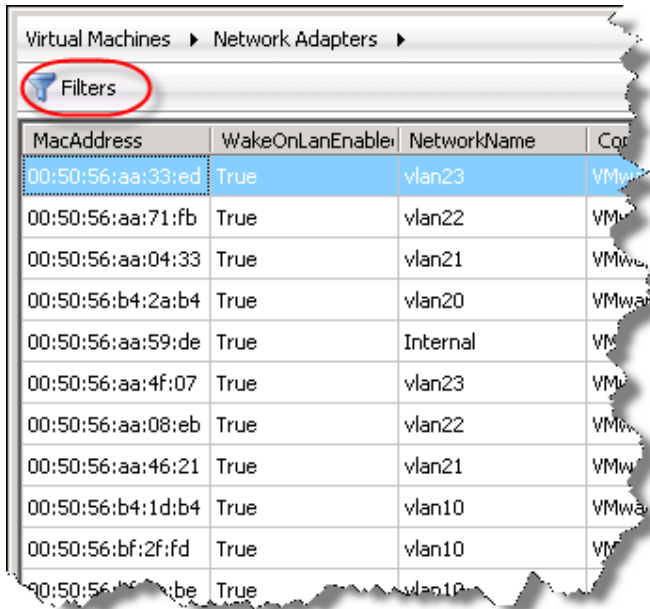
Firstly, select all the virtual machines in the list



Secondly, click the Network Adapters link to the right-hand side

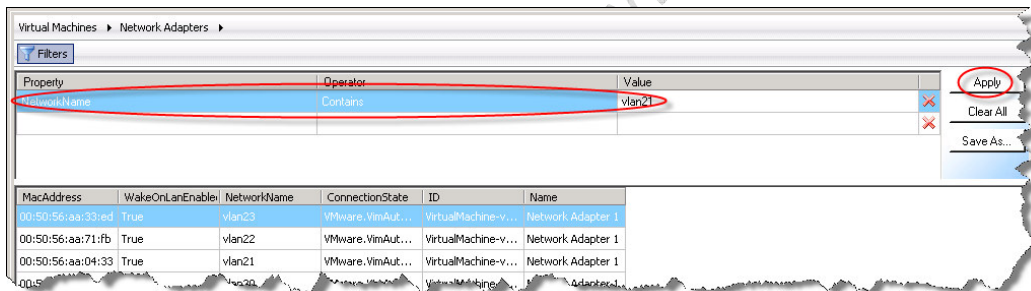


Thirdly, in the new dialog box – click the **Filters** button



MacAddress	WakeOnLanEnable	NetworkName	ConnectionState
00:50:56:aa:33:ed	True	vlan23	VMware.VimAut...
00:50:56:aa:71:fb	True	vlan22	VMware.VimAut...
00:50:56:aa:04:33	True	vlan21	VMware.VimAut...
00:50:56:b4:2a:b4	True	vlan20	VMware.VimAut...
00:50:56:aa:59:de	True	Internal	VMware.VimAut...
00:50:56:aa:4f:07	True	vlan23	VMware.VimAut...
00:50:56:aa:08:eb	True	vlan22	VMware.VimAut...
00:50:56:aa:46:21	True	vlan21	VMware.VimAut...
00:50:56:b4:1d:b4	True	vlan10	VMware.VimAut...
00:50:56:bf:2f:fd	True	vlan10	VMware.VimAut...
00:50:56:bf:2f:fd	True	vlan10	VMware.VimAut...

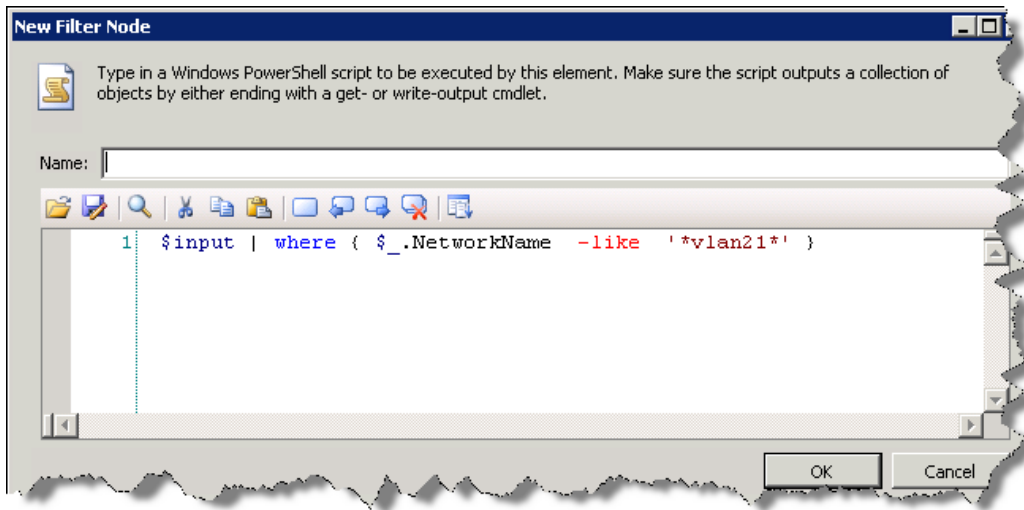
Then create a filter, that search for the portgroup value by the type of NetworkName and click Apply like so:



Property	Operator	Value
NetworkName	Contains	vlan21

MacAddress	WakeOnLanEnable	NetworkName	ConnectionState	ID	Name
00:50:56:aa:33:ed	True	vlan23	VMware.VimAut...	VirtualMachine-v...	Network Adapter 1
00:50:56:aa:71:fb	True	vlan22	VMware.VimAut...	VirtualMachine-v...	Network Adapter 1
00:50:56:aa:04:33	True	vlan21	VMware.VimAut...	VirtualMachine-v...	Network Adapter 1

After clicking the Apply button the UI will only show virtual machine matching your filter. Clicking the Save As... button will show you the PowerShell code for that search like so



Conclusions

As you can see PowerGUI and the PowerShell from VMware is pretty easy to pick up. Sure it doesn't contain everything you might want to do – but the more of us who use it the better the PowerPack from VMware from PowerGui.org will only get better.

If you want to learn more about PowerShell for VMware – a good place to start is the VMware PowerShell blog:

<http://blogs.vmware.com/vipowershell/>

RTFM Education - www.rtfm-ed.co.uk

RTFM Education - www.rtfm-ed.co.uk