

Chapter 10: Site Recovery without VMware SRM

Special Acknowledgement

I would like to give a special acknowledgement to three individuals who directly helped with this section – with specific reference to the PowerShell aspect. I would like in particular to thank Carter Shanklin of VMware, who was happy to answer my direct emails and is the Product Manager for the VMware PowerShell Toolkit.

Additionally, I would like to thank Hal Rottenberg who I first met via the VMware Community Forums. Hal is the author of a new book called “Managing VMware Infrastructure with PowerShell”. If you wish to learn more about the power of PowerShell then I would certainly recommend watching and joining the VMware VMTN community forum – and purchasing Hal’s book.

Lastly, I would like to thank Luc Dekens from the PowerShell forum who was especially helpful in explaining how to create a virtual switch with PowerShell.

Introduction

One of the interesting ironies or paradoxes that came into mind when writing this book was – what if at the point of requiring my DR Plan, VMware’s Site Recovery Manager failed or was unavailable. Put another way, what’s our recovery plan for SRM! Joking apart it’s a serious issue worth considering, there’s little point in using any technology without a Plan B if Plan A doesn’t pan out as we expected.

Given that the key to any recovery plan is replication of data to a Recovery Site – at the heart of the issue the most important element is taken care of by your storage array not by VMware’s SRM. Remember all that VMware SRM is doing is automating a very manual process. So there are really two main agendas behind this chapter – the first how to do everything VMware SRM does manually in case our Plan A doesn’t work out. The second is to show how incredibly useful SRM is in automating this process. Hopefully you will see in this chapter how hard life is *without* VMware’s Site Recovery Manager. Like any automation or scripted process, you don’t really see the advantages until you know what the manual process is like.

With that in mind I could have started with this chapter as chapter 1 or 2, but I figured you would want to get stuck into SRM which is the topic of this book, and save this content for the end. It will also give you an idea of what SRM is doing in the background – which is making your life much easier. The big advantage of SRM to me is that it grows and reacts to changes in your Protected Site – something a manual process would need much higher levels of maintenance to achieve.

As part of the preparation for this chapter – I decided to delete the protection groups and recovery plans associated with our bi-directional configuration from my Recovery Site (Reading)

The manual recovery of virtual machines will require some storage management – stopping the current cycle of replication, and making the “remote volume” a primary volume and read-writable. Whilst SRM does this automatically for you using the SRA from your storage vendor – in a manual recovery you will have to do this yourself. This is assuming you still have access to the array at the Protection Site, as with a planned execution of your DR plan. Additionally, once the replication has been stopped we will have to grant the ESX hosts in the Recovery Site access the last good snapshot that was taken.

On the ESX hosts side – once granted access to the volumes they will have to be manually rescanned – to make sure the VMFS volume has been displayed. Based on our requirements and the visibility of the LUN we will have the option to either not resignature or to force a resignature of the VMFS volume.

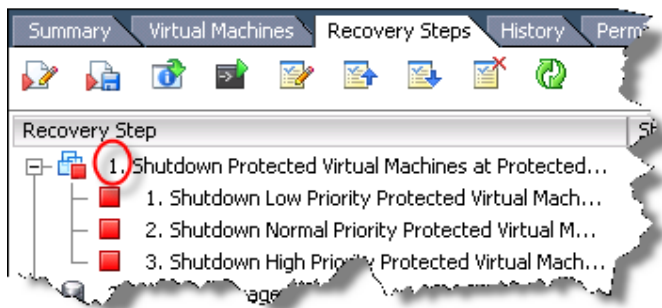
After we have handled the storage side of things we will have to edit the VMX file of each virtual machine and map it to the correct network. After doing that we would then be able to start adding each individual virtual machine into the Recovery Site – each time having to tell the Vi Client which cluster, folder and resource pool to use.

In the ideal world some of virtual machine management could be scripted using VMware's various Software Development Kits such as the Perl Scripting ToolKit, the PowerShell Scripting Toolkit or the VirtualCenter SDK with some language of your choice – VB, C# and so on. I intend to use the PowerShell Toolkit for VMware as an example.

As you will see with scripting the process is laborious and deeply tedious. Critically, it's quite a slow process as well and this would impact on the rapidity of your recovery process. Think of all those RTOs and RPOs.

For a Unplanned Recovery

For unplanned recovery we must shut down the virtual machines that are currently running in production before we manage the storage. If you remember the first steps of any planned recovery is to power down the virtual machines in the Protected Site

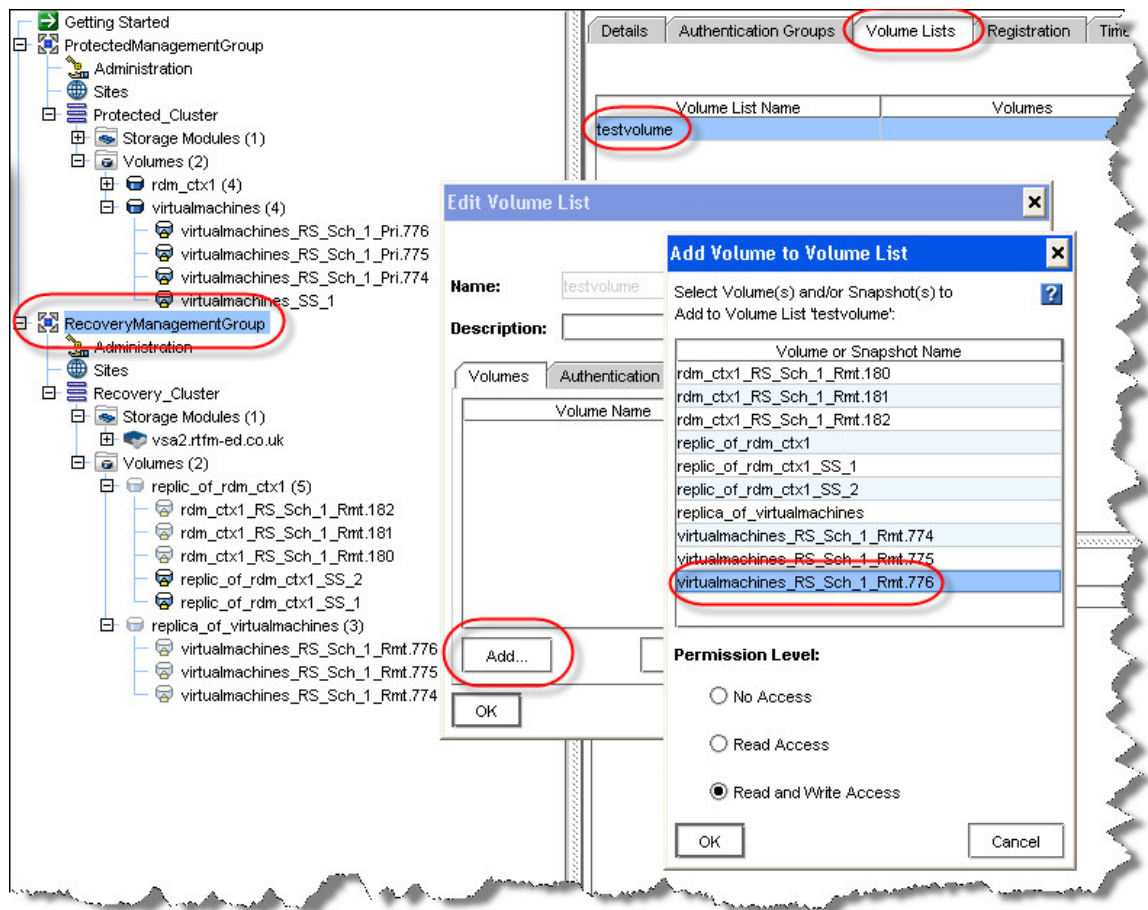


If you are doing a manual failover for testing purposes this is not necessary.

Manage the Storage

Without an SRA, we will have to engage more with the vendor's tools for controlling snapshots and replication. This area is very vendor specific so I would refer you to their documentation. In terms of Lefthand Network's VSA it would be the following steps. Currently, RecoveryManagementGroup (Reading) is holding the primary copy and the ProtectedManagement Group (London) is holding a remote copy maintained by a scheduled remote snapshot. I will need to give the ESX hosts access to the latest replicated or snapshot volume, by adding it to an existing volume list

1. **Login to the Lefthand Networks VSA as administrator**
2. **Select the ProtectedManagementGroup and then Select the Volume Lists tab**
3. **Select an existing volume list and add the latest snapshot to the list**



Note:

In the example above I'm giving my ESX hosts in London access to the last snapshot of my virtualmachines volume (776)

VMware PowerShell Toolkit

The following sections discuss the manual process of getting virtual machines added into VirtualCenter and ready for power on. I've also decided to show you how to do the same tasks with PowerShell. Currently the version of PowerShell I'm working with is a bit bleeding edge – but by the time this book is published it will be publically released.

What follows is almost a getting started guide for getting setup with PowerShell. First of all download and install the Windows PowerShell V2 Community Technology Preview 2 (CTP2). I would put a URL in the book, but it is too long and will ultimately change. Currently the official release of Microsoft's PowerShell is version 1. There will eventually be a version 2 release – currently it's just a "Community Technology Preview".

Next you will need to download and install the VMware PowerShell Toolkit (actually it's called the Vi3 Toolkit), the URL for this is a bit more worthwhile

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

Next you will need to open the PowerShell session, and download into the session the VMware PowerShell Community Extensions.

```
(new-object net.webclient).DownloadString("http://communities.vmware.com/servlet/JiveServlet/downloadBody/6051-102-1-3481/Extensions.psm1") > $env:temp/Extensions.psm1  
add-module $env:temp/Extensions.psm1
```

Once they have been downloaded, you can install them with

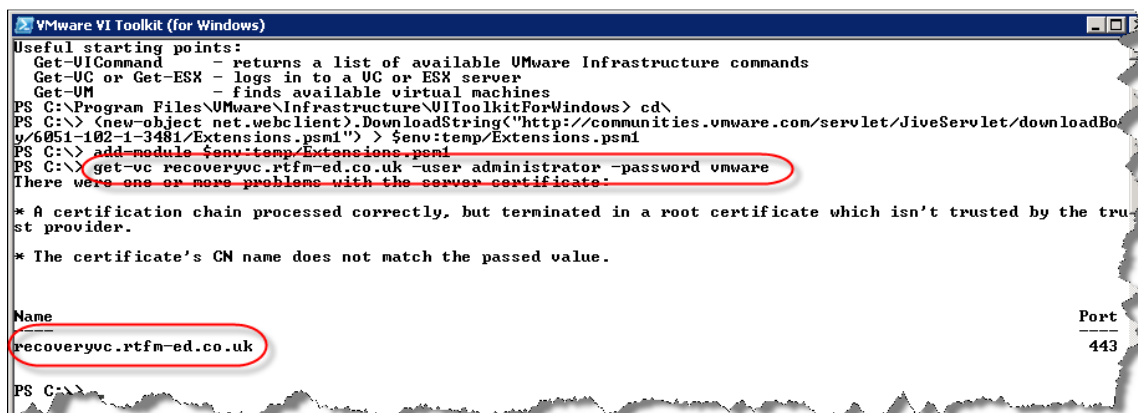
```
add-module $env:temp/Extensions.psm1
```

CAUTION:

I found I had to repeat this process every time I opened a new PowerShell session

To connect and login into the VirtualCenter you use the get-vc command like so:

```
get-vc londonvc.rtfm-ed.co.uk -user administrator -password vmware
```



```
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-VM - finds available virtual machines  
PS C:\Program Files\VMware\Infrastructure\UIToolkitForWindows> cd\  
PS C:\> (new-object net.webclient).DownloadString("http://communities.vmware.com/servlet/JiveServlet/downloadBody/6051-102-1-3481/Extensions.psm1") > $env:temp/Extensions.psm1  
PS C:\> add-module $env:temp/Extensions.psm1  
PS C:\> get-vc recoveryvc.rtfm-ed.co.uk -user administrator -password vmware  
There were one or more problems with the server certificate:  
  
* A certification chain processed correctly, but terminated in a root certificate which isn't trusted by the trust provider.  
  
* The certificate's CN name does not match the passed value.  
  
Name Port  
-----  
recoveryvc.rtfm-ed.co.uk 443  
PS C:\>
```

WARNING:

I imagine this will change before the book is released, and I will try and revisit this just before we go to print to see if this software has been properly released. If you're finding it difficult to find these various software and links – then this is a good place to start – I used it as a starting point to download all the binaries I needed

<http://blogs.vmware.com/vipowershell/2008/06/fun-with-powers.html>

TIP:

I do understand how difficult it is to type all this kind of code by hand, and with this being a paper book, there's no cut and paste options.

So, I have taken all these PowerShell examples and put them into a text file. You can download this text file from the RTFM website so you can cut and paste, and just change the variables such as your virtual machines names and resource pool names.

<http://www.rtfm-ed.co.uk/srm.html>

Rescan the HBAs of each ESX host

You should be more than aware of how do a rescan of an ESX host either from the GUI or CLI. But what you are looking is for a new VMFS volume to appear.

Identification	Device	Capacity	
esx3:storage1	vmhba0:0:0:9	52.75 GB	64...
isos	vmhba1:1:0:1	67.50 GB	17
sanstorage	vmhba1:0:0:1	101.25 GB	9
snap-398cdef9-london_virtualmachines	vmhba32:50:0:1	1,023.75 GB	99
templates	vmhba1:3:0:1	135.00 GB	

This rescan has to be done once per affected ESX host, and it would be quite laborious to do this via the Vi Client (and again, if you made a mistake with the storage layer). Of course you could login with PuTTY and use `esxcfg-rescan`. Personally, I would be more tempted to use the Remote CLI for Windows which could be installed to the same management system as the PowerShell for VMware. With the RCLI for Windows we could use either the `esxcfg-rescan.pl` script

`esxcfg-rescan.pl --server esx1 --username root --password vmware vmhba32`

Whilst the Remote CLI is a nifty tool which will come into its own – once ESX3i (the skinny latte version of VMware’s hypervisor) takes a grip and the Service Console (the full-fat version of the hypervisor) is depreciated or removed altogether – nonetheless it’s not really geared up for bulk administration. This following piece of PowerShell rescans all the ESX hosts in VirtualCenter which is much more efficient from a scripting perspective.

`get-vmhost | get-vmhoststorage -rescanallhba`

```
PS C:\> get-vmhost | get-vmhoststorage -rescanallhba
Softw FileSystemVolumeInfo          ScsiLun
areIS
csiEn
abled
-----
True  {sanstorage, isos, snap-398cdef9... <key-vm.host.ScsiDisk-vmhba1:3:0, ke...
True  {esx3:storage1, sanstorage, snap... <key-vm.host.ScsiDisk-vmhba1:3:0, ke...
```

Note:

The syntax of the above piece of PowerShell is relatively easily to explain. `Get-vmhost` retrieves all the names of the ESX hosts in the VirtualCenter that you authenticated to and this is piped to the command `get-vmhoststorage`, to then rescan all the ESX hosts. `Get-vmhoststorage` supports a `-rescanALLhba` switch which does exactly what you think it does.

You may not or may not find that the VMFS volume doesn’t reflect the original datastore name. Much depends on whether it has been resignedatured either by SRM or manually using the Advanced Settings on ESX hosts. If you want to rename the VMFS volume as SRM does after a resignature you can use this piece of PowerShell

`set-datastore -datastore (get-datastore *london_virtualmachines) -name london_virtualmachines`

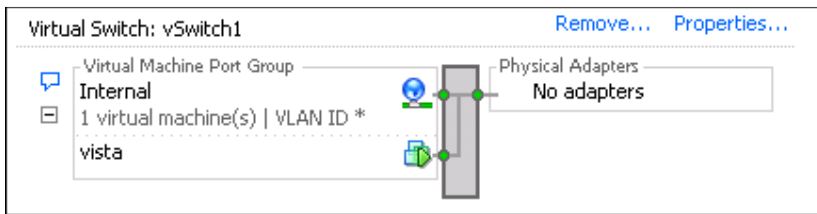
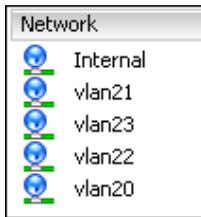
```
PS C:\> set-datastore -datastore (get-datastore *london_virtualmachines) -name london_virtualmachines
Name                FreeSpaceMB    CapacityMB
-----
london_virtualmachines 1015751        1048320
```

Note:

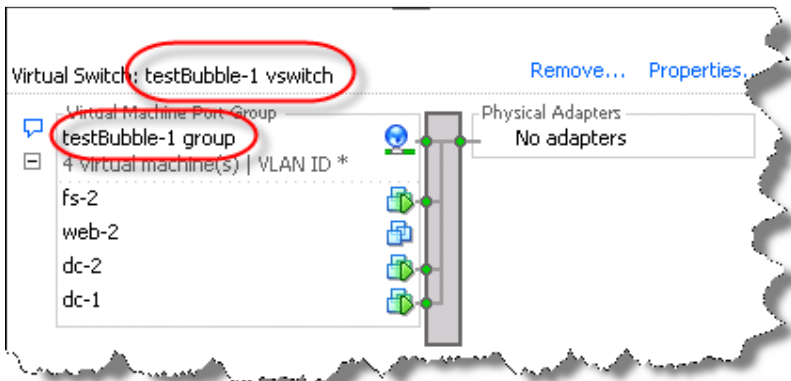
Set-datastore can be used to rename the VMFS volume and datastore name with the - name switch. Used in conjunction with the get-datastore cmdlet, we can search (by wildcard using a *) for the VMFS volume that includes the string "london_virtualmachines", and rename it back to the original VMFS volume and datastore name.

Create an Internal Network for Test

It's part of my standard configuration on all my ESX host that I create a port group called "internal" which is on a dedicated switch with no physical NIC uplinked to it



However, you might wish to more closely emulate the way SRM does its tests of recovery plans which create a "testbubble" network.



Creating virtual switches in the beta release of VMware's PowerShell is quite hard. This has been noted by the VMware PowerShell team, and it's likely to get easier nearer to the GA release. The PowerShell for this is quite lengthy:

```
{noformat}
$MyvSwitchName = "testBubble-1 vswitch"
$MynumPorts = 64
$MyPortGroupName = "testBubble-1 group"
$MyvlanID = 0
```

```
Get-VMHost | %{Get-View (Get-View $_.ID).configmanager.networkSystem} | %{
    $vSwitchSpec = New-Object vmware.vim.HostVirtualSwitchSpec
    $vSwitchSpec.numPorts = $MynumPorts
```

```

$_.AddVirtualSwitch($MyvSwitchName,$vSwitchSpec)

$PortgroupSpec = New-Object vmware.vim.hostportgroupspec
$PortgroupSpec.vswitchname = $MyvSwitchName
$PortgroupSpec.Name = $MyPortGroupName
$PortgroupSpec.vlanID = $MyvlanID
$PortgroupSpec.policy = New-Object VMware.Vim.HostNetworkPolicy
$_.AddPortGroup($PortgroupSpec)
}
{noformat}

```

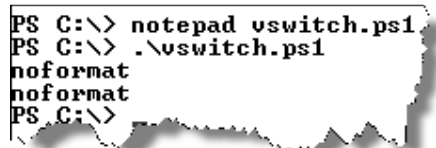
Note:

In this case at the top of the PowerShell script we have a number of variables being defined such as the virtual switch name, number of ports (16,32,64 up to 1024), and the first portgroup name with no VLAN setting. This script does not create a switch with any vmnic's mapped to it. To make this run you would open a PowerShell session and save it to a PowerShell script file (.ps1) with

notepad vswitch.ps1

to execute the script you would type:

.\vswitch.ps1

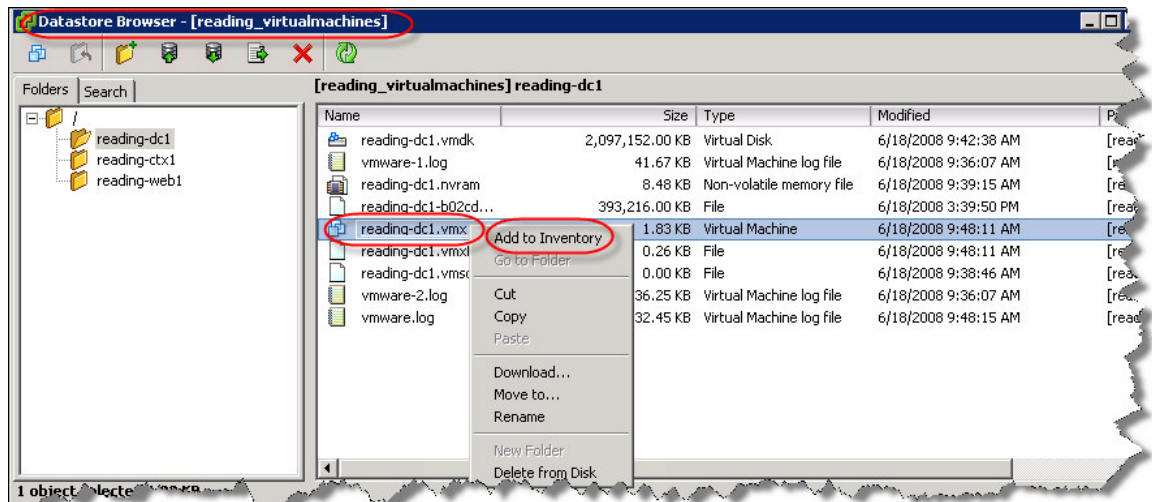


Note:

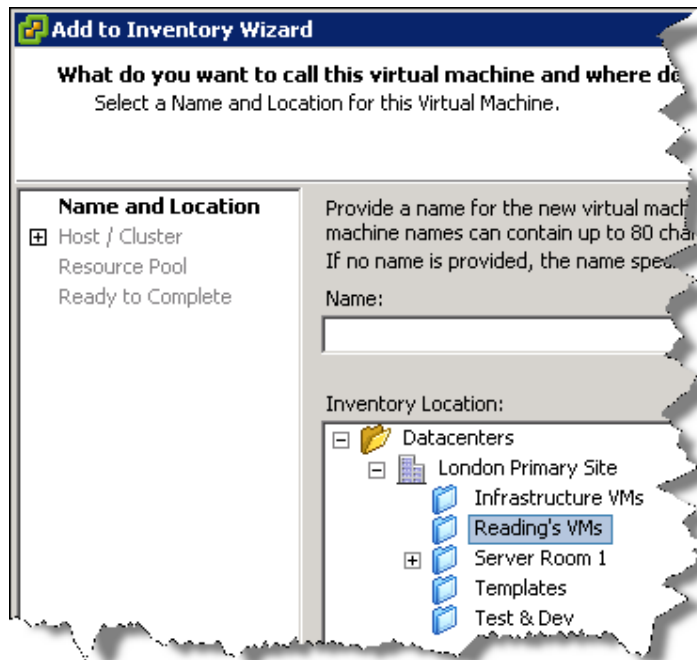
On the PowerShell blog is there is a way of doing this using a XL spreadsheet such that if you are creating Service Console or VMKernel port groups you can pipe unique IP address for each ESX hosts you run the script on.

Add Virtual Machines to the Inventory

1. On one of the **ESX hosts**
2. **Browse the data store** that contains the relevant virtual machines
3. **Right-click the VMX file** and **Choose Add to Inventory**



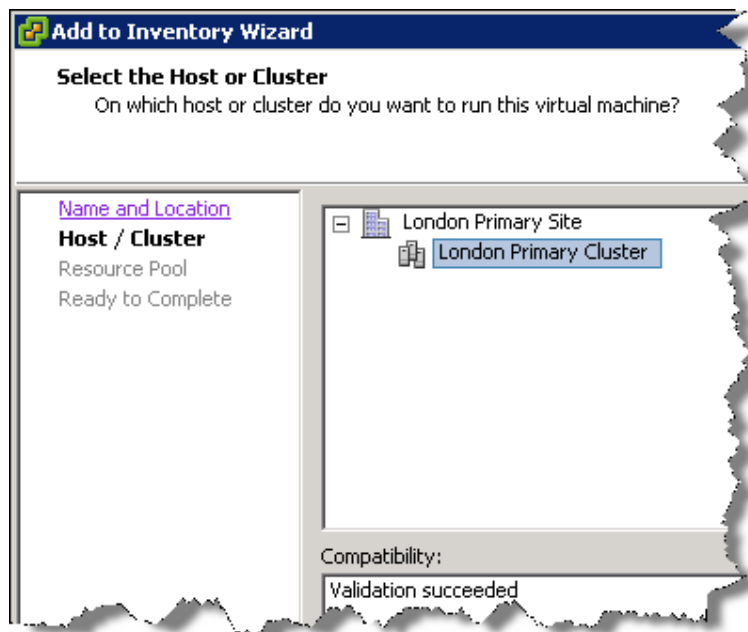
4. In the **subsequent** dialog box **select a DataCenter and Folder** for holding your virtual machine



Note:

You don't have to specify a name for your virtual machine, this causes VirtualCenter to read the VMX file for the displayName = "..."

5. **Select the ESX Host or Cluster**



6. Next **select a resource pool** for the virtual machine



Note:

You should now be able to power on the virtual machine. You will have to manually change its IP address in the guest operating system. Its possible to automate adding a virtual machine to a ESX host (not a cluster) using the command-line ESX host tool called vmware-cmd. Unfortunately, it cannot handle the metadata of VirtualCenter – such as folder location and resource pool.

Remember you would have to repeat these steps for each and every virtual machine needed to be recovered.

Perhaps a better way is to use some PowerShell. We can use the cmdlet get-datastorefile and get-datastore to provide a list of all the virtual machines on the datastore

get-datastorefiles (get-datastore london_virtualmachines) | where { \$_.Path -match '.vmx\$'} | select path

```
PS C:\> get-datastorefiles (get-datastore london_virtualmachines) | where { $_.Path -match '.vmx$'} | select path
Path
-----
[london_virtualmachines] ctx-1/ctx-1.vmx
[london_virtualmachines] ctx-2/ctx-2.vmx
[london_virtualmachines] fs-1/fs-1.vmx
[london_virtualmachines] fs-2/fs-2.vmx
[london_virtualmachines] web-1/web-1.vmx
[london_virtualmachines] web-2/web-2.vmx
[london_virtualmachines] dc-1/dc-1.vmx
[london_virtualmachines] dc-2/dc-2.vmx
[london_virtualmachines] sql-1/sql-1.vmx
[london_virtualmachines] sql-2/sql-2.vmx
```

Note:

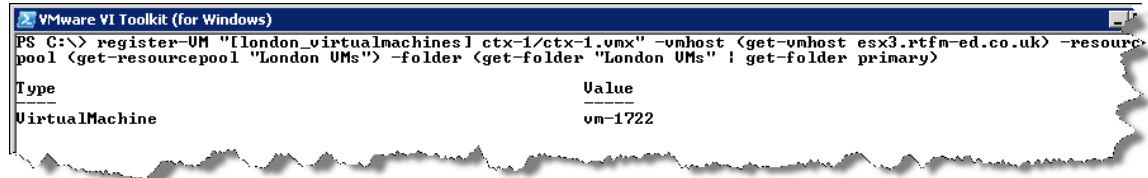
This piece of PowerShell lists all the folders/files (get-datastorefiles) on a given VMFS volume which is retrieved by the get-datastore cmdlet. This is then filtered through a search (select path) to only show path information where a folder contains a VMX file. Finally, the output is filter again – just to show file paths in this format

[london_virtualmachines] /ctx1/ctx1.vmx

Once we have the path we can think about trying to register a VM , there is a Register-VM cmdlet which we can use to handle the full registration process including esx host, folder, and resource pool location in the VirtualCenter inventory like so:

register-VM "[london_virtualmachines] ctx-1/ctx-1.vmx" -vmhost (get-vmhost

esx3.rtfm-ed.co.uk) -resourcepool (get-resourcepool London) -folder (get-folder "London VMs" | get-folder primary)



```
PS C:\> register-VM "london_virtualmachines1 ctx-1/ctx-1.vmx" -vmhost (get-vmhost esx3.rtfm-ed.co.uk) -resourcepool (get-resourcepool "London VMs") -folder (get-folder "London VMs" | get-folder primary)
```

Type	Value
VirtualMachine	vm-1722

Note:

Hopefully most of the above makes sense. The reason I have -folder (get-folder "Reading VMs" | get-folder primary) is that I have two folders, one in London VMs and one in Reading VMs. If I just searched get-folder primary – it would report two MOREF (Managed Object Reference Numbers) and Register-VMs would not know where to place the virtual machine in the inventory – and an error would occur.

This output can be redirected to the Register-VMs cmdlet to register all the virtual machines found on that datastore like so:

```
$vmxpath = get-datastorefiles (get-datastore reading_virtualmachines) | where { $_.Path -match '.vmx$' } | select path | % { Register-VM $_.Path -vmhost (get-vmhost esx3.rtfm-ed.co.uk) -resourcepool (get-resourcepool "London VMs") -folder (get-folder "London VMs" | get-folder primary) }
```

Note:

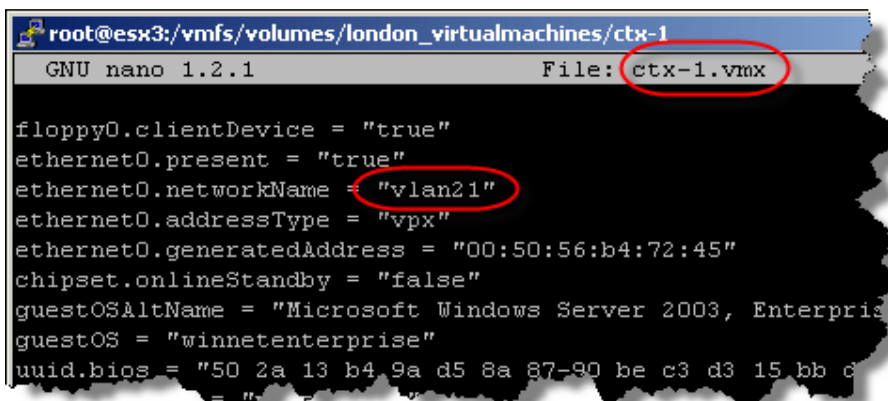
The only difference here is the inclusion of \$vmxpath and the %. In this case we're making the path to the VMX files a variable, and then piping it to the Register-VM command so then every virtual machine (VMX file path) discovered in the reading_virtualmachines VMFS is then registered to the system

Remember this registration process would have to be repeated for each and every VMFS volume, for each and every VM needing recovery. I'm assuming that every VM in a VMFS should be registered with the ESX host.

Fix VMX Files

Using nano or Vi at the Service Console edit the vmx file of each virtual machine to fix the port group used for communication.

ethernet0.networkName = "vlan61" to be ethernet0.networkName = "internal"



```
root@esx3:/vmfs/volumes/london_virtualmachines/ctx-1
GNU nano 1.2.1 File: ctx-1.vmx
floppy0.clientDevice = "true"
ethernet0.present = "true"
ethernet0.networkName = "vlan21"
ethernet0.addressType = "vpx"
ethernet0.generatedAddress = "00:50:56:b4:72:45"
chipset.onlineStandby = "false"
guestOSAltName = "Microsoft Windows Server 2003, Enterprise Edition"
guestOS = "winnetenterprise"
uuid.bios = "50 2a 13 b4 9a d5 8a 87-90 be c3 d3 15 bb c"
```

Alternatively, repeat this for each virtual machine behind

If you add your virtual machine into the VirtualCenter first (our next task) you can then automate the property change (as I mentioned previously) with PowerShell for VMware.

```
get-vm | get-networkadapter | sort-object -property "NetworkName" | where {'vlan21' -contains $_.NetworkName} | Set-NetworkAdapter -NetworkName "testBubble-1 group"
```

```
PS C:\> get-vm | get-networkadapter | sort-object -property "NetworkName" | where {'vlan21' -contains $_.NetworkName} | Set-NetworkAdapter -NetworkName "testBubble-1 group"
Confirm
Are you sure you want to perform this action?
Performing operation "Setting NetworkName: testBubble-1 group" 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:aa:46:21
WakeOnLanEnabled : True
NetworkName     : testBubble-1 group
ConnectionState : VMware.VimAutomation.Client20.ConnectInfoImpl
ID              : VirtualMachine-vm-1742/4000
Name            : Network Adapter 1

MacAddress      : 00:50:56:aa:18:43
WakeOnLanEnabled : True
NetworkName     : testBubble-1 group
ConnectionState : VMware.VimAutomation.Client20.ConnectInfoImpl
ID              : VirtualMachine-vm-1732/4000
Name            : Network Adapter 1

MacAddress      : 00:50:56:aa:08:eb
WakeOnLanEnabled : True
NetworkName     : testBubble-1 group
ConnectionState : VMware.VimAutomation.Client20.ConnectInfoImpl
ID              : VirtualMachine-vm-1744/4000
Name            : Network Adapter 1
```

You could at the service console use the 'sed' command to find and replace the port group string inside the VMX file, but I think the PowerShell method is neater. Again this piece of Powershell use get-vm and get-network to find every virtual machine and the portgroup setting – it's then sorted and filtered to only show virtual machines whose portgroup is vlan61. Once this filter has been applied it is then piped to the set-networkadapter cmdlet which adjusts all the virtual machines with vlan61, and replaces it with a portgroup value called internal.

Conclusions

As you can see the manual process is very labor intensive – which is to be expected by the use of the word manual. You might have got the impression that this issue can all be fixed by some wizz-bang PowerShell scripts. You might have even thought – sucks, why do I need SRM if I have these PowerShell scripts. However, it's not as simple as that for two main reasons – there's no support really for this home-brewed DR – and secondly, you can work all you like testing your scripts – but then your environment will change – and those scripts will go out of date and will need endless re-engineering and re-testing.

In fact the real reason I wanted to write this chapter is to show how painful the manual process is – to give you a real feel of the true benefits of SRM.

