VMware® Infrastructure 3

Advanced Technical Design Guide

~and~

Advanced Operations Guide

Two books in one!

Ron Oglesby
Scott Herold
Mike Laverick

Chapter 3: Storage

In this module we will be looking at the three primary ESX 3.x storage options – Storage Area Network (SAN), Internet SCSI (iSCSI), and Network Attached Storage (NAS). Additionally, we will take a look at VMware's File System (VMFS) and explain how to format and extend a VMFS volume.

By their nature SAN and iSCSI systems are very proprietary. My intention here is not to teach about the configuration of HP, IBM, or EMC storage, but to explain and demonstrate how VMware can leverage this storage.

Configuring SAN based storage

One of the most common tasks required of an ESX administrator is finding out their World Wide Name (WWN). This is a 64-bit address, stamped on every Host Bus Adapter (HBA) that has a connection to the SAN. The reason storage people request this information is that they use this address to allow or deny access to LUNs on the SAN. You can see that the WWN is akin to the MAC address on a network card.

It is extremely unlikely that anyone will buy a SAN and only use it with ESX server. It is very likely that existing physical servers running an array of very different OS would have access to the same SAN. Since none of the following share a common-file-system, corruption of data is possible: ESX, Linux, Novell, Windows, and Solaris.

SAN administrators use the WWN to "mask," or hide, these LUNs to make sure the right OS sees the right LUNs. Usually the SAN vendors will have their own product specific term for this feature. For example, HP calls their technology "Presentation" or "Selective Presentation." If your SAN does not support firmware-based "LUN Masking" then you might be interested to know that ESX server can mask the LUNs through an ESX server configuration setting (Configuration Tab, Software Pane, Advanced Settings and setting called Disk.MaskLUN'S). This said, most production SANs will support LUN mask-

ing and this should always be preferred over and above any other software method.

As an ESX administrator, you might be required to provide the WWN of your server or servers so that storage people will be able to set this up. Alternatively, if you are fortunate enough to manage your own SAN you may require this information personally.

Lastly, some organizations go that extra step further. Alongside LUN masking they might also create "zones," similar to Ethernet's VLAN, for these SAN switches. It's a way of configuring the switch into smaller networks. So, not only is a server restricted by its WWN, it would also need to be plugged into the correct port within the correct zone. Additionally, this zoning can offer some performance benefits to the fabric.

Finding out your HBA's WWN

As with network cards, all storage adapters are labeled vmhba0, vmhba1 and so on. Usually the internal RAID controller card has a HBA number of 0 because this naming convention merely looks at the PCI bus and serializes the storage adapters as they are found during the installation. So in most cases, your first fiber channel HBA is likely to be vmhba1 and so on.

4. Select your ESX host.
5. Click the Configuration Tab.
6. In the Hardware Pane choose Storage Adapters.

Rescanning SAN LUNs

After new LUNs have been made available, or the storage people have done their work, you will need to force a rescan of the HBA from the VI Client. On boot-up, ESX scans all the LUNs it is able to see – but you really don't want to do unnecessary reboots of the ESX server itself just to see a new LUN. This is why the rescan feature exists.

ESX server can see up to a maximum of 256 LUNs. That's LUN 0 to LUN 255. In ESX 2.x there was a setting called Disk.MaxLUN that would stop this scanning once an ESX host had scanned from LUN0 to LUN7. This value has now been changed, mainly to reduce the number of support requests generated by VMware customers not knowing about the Disk.MaxLUN value.

The settings for Disk.MaxLUN have now been changed to 256 with a starting and scanning position of 1. On some systems, LUN0 is a management LUN and should not be used. You should consult your documentation if you are not sure.

The rescan dialog box allows you to rescan for new LUNs and also check if they contain VMFS volumes. To force a rescan:

1. Select your ESX host.
2. Click the Configuration Tab.
3. In the Hardware Pane.
4. Under Storage Adapters.
5. Select the HBA you wish to rescan,
6. Click the Rescan… from the top right-hand corner of the VI Client.
7. Click OK to the dialog box.

Note:

You can also force a rescan using the command-line tool esxcfg-rescan. Its parameters are esxcfg-rescan vmhba*N*, where vmhbaN is the host bus adapter used for the rescan process. Its ability to show you the LUN discovery process in more detail can be useful.

Understanding the vmhbaA:T:L:V syntax

ESX server uses a special syntax so that the VMkernel can find its "path" to the storage it is using. This is used by local storage, SAN, and iSCSI. The syntax of vmhba*N:N:N:N* is a sequence of numbers which tells the VMkernel how to

navigate to the storage location in question. This syntax is not unique to ESX server; it has been in used in UNIX for many years.

It is best remembered by the acronym A:T:L:V which stands for Adapter: Target: LUN: and Volume. The target in SAN or iSCSI environment represents the storage processors (SP) on the disk array. The first three digits (A:T:L) begin their numbering at 0, whereas the last digit (V) starts with 1.

For example, vmhba2:1:18:3 would tell me I was using the third HBA (0,1,2) connected to the second storage processor (0,1) using the LUN 18 – and the volume or partition I was dealing would be the third partition (3).

Note: Remember LUN 18 is actually the 19th LUN, as we start counting from 0.

In day-to-day operations you usually won't need to know this syntax. Most people use volume and datastore labels which are available when you format a LUN as VMFS. However, you will need to know this syntax when formatting VMFS volumes and to configure other advanced storage options such as Raw Device Mapping (RDM) files. These RDM files allow the ESX administrator to give a VM direct access to LUNs on a SAN or iSCSI system. We will look at RDM files in Chapter 5 when I cover creating and modifying VMs.

iSCSI

iSCSI is a competitor to SANs. This said, frequently the people who make SANs make iSCSI equipment, too. iSCSI is very similar to SAN technology in that it is capable of presenting a LUN to server. What makes iSCSI different is the transport used to carry the SCSI commands. It uses TCP port 3260 to carry out this communication. So, watch out if you work in a restrictive firewall environment, as this port may not be opened.

iSCSI uses your normal Ethernet network infrastructure to carry the commands themselves. This means you don't necessarily need special HBAs and special switches as conventional network cards and switches will do just as well, although it is possible to buy iSCSI HBAs which are optimized for performance.

This also allows you to leverage your existing IP knowledge and equipment. In the future, because iSCSI is, by definition, "internet aware/ready," replicating data from one location to another might be much easier to achieve than it currently is with fibre channel networking.

You can use conventional network cards to connect to an iSCSI system. VMware refers to this as the "Software Initiator." VMware's software iSCSI is actually part of Cisco's iSCSI Initiator Command Reference. The software initiator inside the VMkernel works together with a vmkiscis daemon in the Service Console environment.

Alternatively, you can buy an adapter that is designed for iSCSI communication, which VMware refers to as a "Hardware Initiator" from companies like Qlogic. As the name suggests, an initiator is the side that begins the communication (the client) to the iSCSI target (the disk array). Hardware initiators have what is called a TCP Off-Load Engine (TOE chip) which improves performance by removing the load from the main CPUs. In the future, all systems will have a TOE chip which will be enabled/disabled in the BIOS. In fact, the new HP G5s already supports this multi-functionality.

Another unique iSCSI feature is its use of a different naming convention. iSCSI does not use WWN; instead it uses what is called iSCSI Qualified Name (IQN). We can regard the IQN as a DNS name, or reverse DNS. It's a convention rather than something which is hard-coded. IQN would look something like this:

iqn.2006-11.com.vi3book:iscsi1

The first part is always the text IQN followed by the domain name registration date, and ending in the name itself. In reality, it is only used to ensure uniqueness. After all, a domain name can't be registered by more than one organization at the same time. The next part is your domain in reverse. For us, vi3book.com would be com.vi3book or uk.co.rftm-ed or com.vmguru. After that, you can specify a colon and alias; in the example, the alias has been set to iscsi1. As with all aliases, the intention is allow you to use short names instead of specifying the full IQN in software interfaces.

With the software adapter the IP address, subnet mask, and default gateway are specified in a VMkernel port group on vSwitch. The IQN itself is specified when you configure the software adapter for the first time. If you are working with a hardware adapter such as a Qlogic iSCSI card, you will probably find these settings are held in the cards BIOS settings.

You will generally want to have an isolated network for your iSCSI traffic – not only for performance but also because the traffic generated, as in SANs, is not currently encrypted. VMware's iSCSI implementation does support authentication with the Challenge Handshake Authentication Protocol (CHAP) for an additional tier of security.

While iSCSI can do all the things SANs do, you should be aware of four current limitations in terms of VMware Support. Firstly, you can install ESX to iSCSI LUN for iSCSI booting purposes, but you must use a supported hardware initiator. Secondly, only the hardware initiator supports a "static discovery" of LUNs from the iSCSI system. This is not a great limitation as "dynamic discovery" is much easier to setup and works with both software and hardware initiators. Thirdly, there is no support for running clustering software within a VM using iSCSI storage. Lastly, there is currently no support from VMware for iSCSI used in conjunction with VMware's Consolidated Backup.

In this section we will focus on the use of the software adapter, as this is a method everyone can configure.

iSCSI Virtual Appliances

What is a virtual appliance? A virtual appliance is a downloadable VM which already contains an OS supported by virtualization vendor which contains a ready-to-run application or service. Sometimes these are free, especially when they are based on Linux; other times they are locked and are only used to evaluate an appliance before parting with hard cash.

There are a number of free appliances available which will emulate an iSCSI system for you. While I wouldn't recommend these for a production environment they are perfectly fine for your test and development labs or for your own personal development in the process of learning VMware, perhaps as part of

your preparation for the VCP test. As with all things in life, book learning is one thing but hands on experience is quite another.

Dave Parson, from Manchester in the United Kingdom, who is active on the VMware Community Forums, has put an iSCSI emulator together as a downloadable virtual appliance. It is downloadable from these locations:

Primary Location:

http://itservices.ne-worcs.ac.uk/pub/vmware/iscsitarget.zip

Alternative Mirrors:

http://chaz6.com/static/files/vmware/hosted/iscsitarget.zip

http://chaz6.com/static/files/vmware/hosted/iscsitarget.md5

Alternatively, you might want to look at OpenFiler.com, a Linux build designed for storage which includes iSCSI support.

If you are interested in iSCSI Virtual Appliance, "iSCSI Virtual SAN" is available from the VMware Appliance Directory web site and has been created by the scarily named user, "reaper007."

http://www.vmware.com/vmtn/appliances/directory/364

Configuring iSCSI based storage

In this section, I will configure software-based iSCSI emulator, or target. After all to, configure ESX initiators there has to be something to communicate to. I would like to say that using particular software does not amount to a recommendation. I merely use them because they are easy to get.

Setting up iSCSI with Fedora Core 5 and iSCSI Enterprise Target (IET)

iSCSI Enterprise Target (IET) is a popular iSCSI emulator written for Linux and is freely distributed under the terms and conditions of a General Public License (GPL). If you prefer to use a Windows platform instead, you can get a free 15-day evaluation copy of "StarWind for Windows," produced by a company called RockDivision.com. If this is the case, you can skip this section and go to the section on setting up with StarWind.

Ideally, you will set this up on a physical system, but if you are just doing this for evaluation purposes a virtual machine will suffice. I have created a VM-based iSCSI box running on ESX host. This ESX host can then connect to the VM iSCSI box and you can format as VMFS. So my VM offers up VMFS partitions to the same ESX host on which it resides! This is a bit crazy I know – but it just goes to show how flexible VMware's virtualization technology can be.

iSCSI will present a LUN to ESX just like a SAN will, so I would recommend at least one LUN for your boot disk and two or more disks to act as iSCSI Targets.

I would like to thank Geert Baeke of baeke.info and Anze Vidmar of fedoranew.org respectively for their online help in this documentation – I "borrowed" extensively from both of these links.

You can download iSCSI Enterprise Target from this location**:**

http://sourceforge.net/project/showfiles.php?group_id=108475

Further documentation is also available on IET's Wiki page:

http://iscsitarget.sourceforge.net/wiki/index.php/Main_Page

1. Download and Install Fedora Core 5

   Note:

   Do a standard installation of Fedora Core 5 without the GUI front-end for Linux.

2. Make sure you enable "Software Development." We will need tools like CC to compile iSCSI Target for our kernel.

3. If you do enable Fedora Core 5's firewall, you will have to enable TCP Port 3260. You can change the configuration of the firewall after the install with the setup command.

4. Confirm your current kernel release with uname –r.

5. Download and update your kernel withyum –y update kernel kernel-devel.

   Note:

   Yum is an automatic updater and package installer/remover for the Redhat Package Management (RPM) systems. It automatically computes dependencies required to install packages. It makes it easier to maintain groups of machines without having to manually update each one using RPM. The –y chooses "yes" to all the prompts that occur. The kernel-devel downloads a development based version of the kernel which we can use during the compilation process.

6. Confirm the prompts from yum, and then reboot to make sure your updated kernel is in use.

7. Check your kernel release again with: uname –r.

8. Using the kernel name derived from above, download the latest kernel sources with yum –y install kernel-devel-*kernel-release-value*.

   Note:

   In my case this was yum install kernel-devel-2.6.17-1.2139_FC5

9. Yum will download the kernel source code and put into the /usr/src/kernels directory

10. Next we will **download and untar the iSCSI Target software** to the /usr/local directory (from http://iscsitarget.sourceforge.net/).

    Note:

    If you want to download directly to your iSCSI box, you could use a mirror hosted at Kent University in the UK with

    ```
    cd /usr/local
    ```

```
wget
http://kent.dl.sourceforge.net/sourceforge/iscs
itarget/iscsitarget-0.4.13.tar.gz

tar xvzf iscsitarget-0.4.13.tar.gz and then cd
iscsitarget-0.4.13
```

11. **Compile and Install** the iSCSI Target Software with

    ```
    export KERNELSRC=/usr/src/kernels/kernel-
    release-value

    make && make install
    ```

    Note:

    In my case the export command was:

    export KERNELSRC=/usr/src/kernels/2.6.17-1.2139_FC5

    If you get an error stating that CC was not found this is because you did a standard install of Fedora Core 5 which doesn't include programming tools. CC is a program that allows you to compile programs from a 'C' programming language/environment

12. **Copy the sample configuration file of iSCSI-Target** to the **/etc directory**

    ```
    cp iscsitarget-0.4.13/etc/ietd.conf /etc
    ```

13. **Edit the ietd.conf** file with

    ```
    nano –w /etc/ietd.conf
    ```

    Note:

    This is quite a lengthy file that has verbose information to explain the syntax. From this file we can specify an iSCSI Target ID. It allows you to set incoming and outgoing authentication and an alias for the target itself. However, ESX only supports incoming authentication to the iSCSI Target (ESX host authenticates to the iSCSI target), not outgoing authentication (iSCSI Target authenticating to the ESX host).

14. I have two LUNs to make available without authentication. So I configured the ietd.conf in the following way:

```
Target iqn.2006-11.com.vi3book:iscsi1
# Users, who can access this target
# (no users means anyone can access the target)
#IncomingUser
#OutgoingUser
# Lun definition
# (right now only block devices are possible)
Lun 10 Path=/dev/sdb
Lun 11 Path=/dev/sdc
# Alias name for this target
Alias iSCSI
# various iSCSI parameters
# Not in use…
```

15. To **Start the Service** and to **configure it to start on boot-up** use:

```
service iscsi-target start
chkconfig iscsi-target on
```

Setting up iSCSI with Windows 2003 and StarWind

Perhaps you would prefer to use Microsoft Windows as the basis of your iSCSI emulator or appliance. As I said before, this would require you to buy the Storage Server edition of Windows 2003 from Microsoft or take an existing copy of Windows and buy a third party software add-on that would make Windows function as an iSCSI Target. (One such third-party company is RockDivision.com. They have a software product called StarWind. A free 15-day evaluation of their product lets you play with it before parting with hard cash.)

Firstly, you need to setup on Windows with more than one disk or LUN. These other LUNs do not need formatting with NTFS because they will be presented to ESX as raw storage. Next, register with Rockdivision to download their StarWind product and copy it to the Windows server you created. The software is easy to install as it is a "next, next, and next" style setup program.

Below is a very simple configuration of the software suitable for our purposes:

1. Allow the installer to **Launch Starwind**.
2. **Double-Click the StarWind icon** on your desktop.
3. Right-Click **local-host:3260** and choose **Connect…**

Note:
If you are using the free evaluation, the password for the test account is: **test**. The evaluation software ships with a RAM drive automatically configured. If you wish you can right-click this and remove it.

4. Click the **Add Device** icon.
5. Choose **Image File device**.

   Note:

   If you buy this product you might prefer SPTI mode. It stands for "SCSI Pass-Through Interface." This feature device allows a hard disk, LUN, or CD-ROM to be shared as an iSCSI Target with the StarWind product. Starwind can run inside a VM but only with the "Image File" method as the VM's controllers (buslogic and lsilogic) are not fully supported.

6. In the "**Image File Device Parameters**" dialog box click the **New Image** button.
7. In the new dialog box, **specify a name and path** such as **e:\disk1.img** and **specify a size such as 10240MB** – and click **Next**.
8. **Enable** the option that reads "**Allow Multiple Connections**."

   Note:

   We can use the multiple connections settings to allow more than one ESX host to connect to the same iSCSI LUN. You can cycle around this process as much as you like creating as many IMG files as you wish.

Setting a VMkernel port group for IP Storage

In the networking section we set up a VMkernel port for VMotion as an example. Here we are going to set up a vSwitch with VMkernel port group for IP storage with the specific aim of using it to communicate with our iSCSI systems. For clarity, I have setup my two iSCSI systems using 172.168 addresses. iSCSI1 is based on Fedora Core 5 and IET with the IP address of 172.168.2.210, and iSCSI2 is based on Windows and StarWind with the IP address of 172.168.2.211.

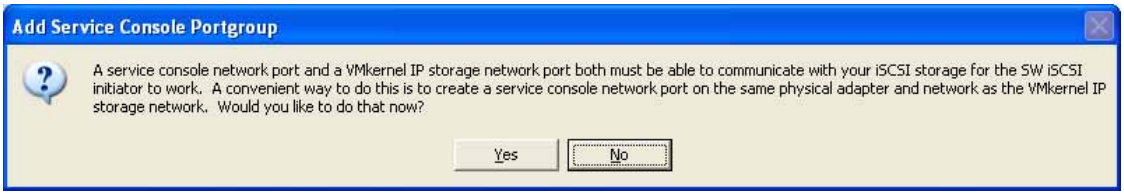
When configuring the networking for the software initiator, you actually allocate two IP configurations. In the vSwitch for iSCSI we configure two port groups; a VMkernel storage port group and Service Console port group. This "second" service console port group is required for using iSCSI. The configuration generally runs smoother if you have already created the second Service Console port first. This leads to less warnings and pop-up dialog boxes.

Figure 3.1 shows a default dialog box that always appears the first time you enable the iSCSI initiator if you don't *first* create the second Service Console port group. If you have already created the second Service Console port this dialog box does not appear.

I will allocate 172.168.3.101/16 to my VMkernel storage port group and 172.168.3.111/16 to my second Service Console port group.

If we were using a hardware initiator we wouldn't have to do this. For example, with Qlogic iSCSI adapters the IP address, Subnet Mask, and Default Gateway values are set in the adapter BIOS using [Alt]+[Q] to access it. If you are using ordinary network cards, as I am, you will have to create a VMkernel port group to set these values.

Figure 3.1



1. Select your ESX host.
2. Click the Configuration Tab.
3. In the Hardware Pane, select Networking.
4. Click the Add Networking… link.
5. Choose VMkernel and Click Next.
6. Click Next.

7. In the Port Groups Properties dialog, type a friendly name for this connection, such as ipstorage.

8. Type a valid IP Address and Subnet Mask for the VMkernel port group.

   Note:

   You will get a message about setting a default gateway entry for the VMkernel port group. This pop-up appears with every VMkernel port group you create if you don't choose "Yes" and set this. It can be little bit irritating. Of course, if your iSCSI systems were behind a router, you would choose "Yes" and add the default gateway entry. I often set the default gateway to be one of the iSCSI or NAS nodes on my network. This is helpful if I am on a router-fewer network and wish to stop the warning dialog box appearing. I can also use this to test my VMkernel IP configuration using the VMware ESX command called vmkping. Of course, a really easy way to check that you can communicate from ESX to your iSCSI system would be to start the ping from the iSCSI system to VMkernel storage port IP address just configured. Next we will add the second Service Console port.

9. Open the properties box for the newly created vSwitch and click the "Add" button to add a second Service Console port group.

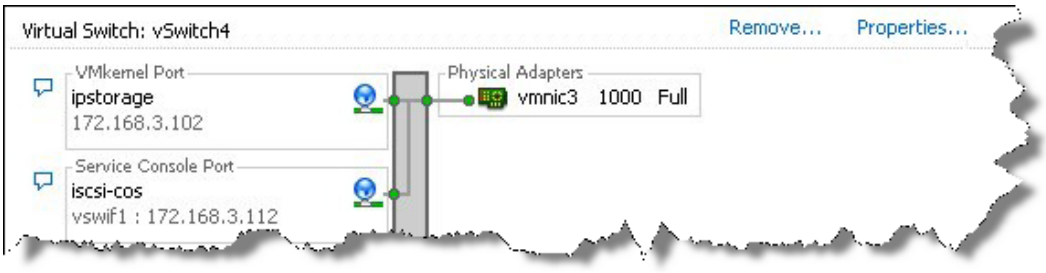
10. Choose "Service Console" as the port group type.

11. Change the port group name to something meaningful, like iscsi-cos.

12. Set a second IP address and Subnet Mask valid for the iSCSI network.

Figure 3.2 shows how my networking has changed to reflect I now need to access the iSCSI storage system.

Figure 3.2



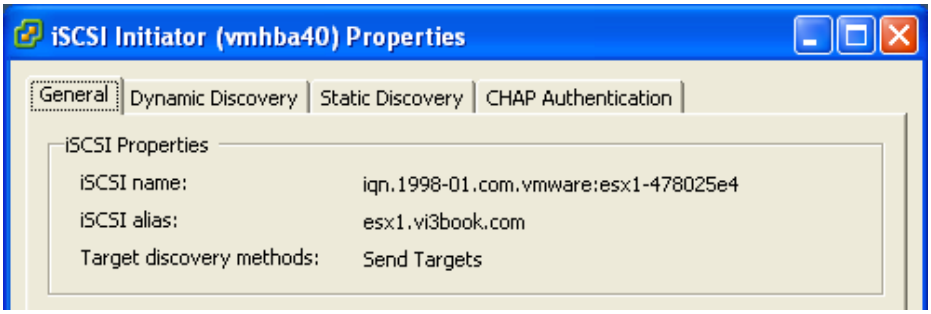
Connecting ESX to iSCSI (Software Adapter):

1. Choose the ESX Host from the list, and select Configuration Tab.
2. In the Hardware Pane, Choose Storage Adapters.
3. Select iSCSI Software Adapter and Properties…
4. In the General Tab, click the Configure button.
5. Select Enable and Click OK.

Figure 3.3 shows how the IQN data will be updated to show that the initiator is enabled and it will automatically generate a unique iSCSI Initiator (iqn.1998.01-com.vmware:esx1-1478025e5) and Alias (esx1.vi3book.com) for you.

If you prefer to use your own naming convention you can click the configure button and set your own IQN values. I usually change this after I have made the iSCSI system setup with ESX because, in some releases of ESX 3, I have seen that VI Client prompts you to do a reboot of the ESX host and I like to avoid reboots if at all possible.

Figure 3.3



1. Select the Dynamic Discovery tab and Click the Add button – type in the IP Address of your iSCSI Target.

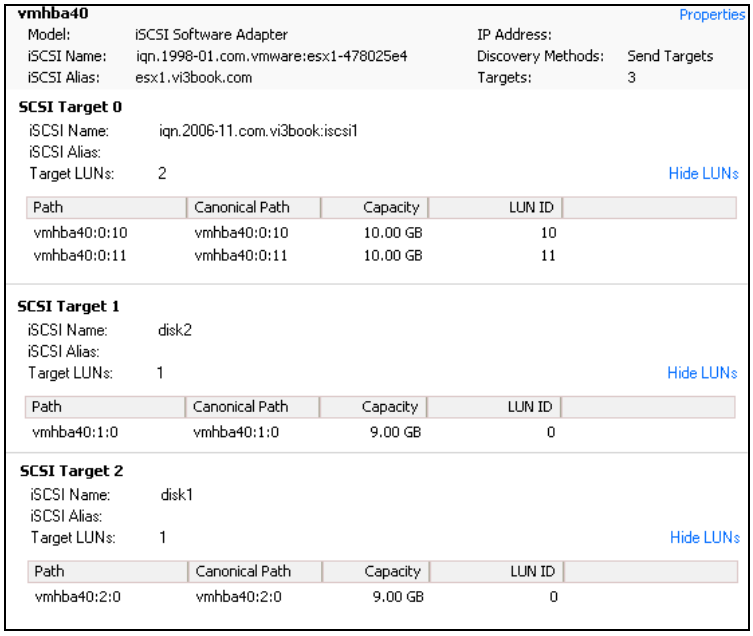
   Warning:

   It can take some time for these dialog boxes to refresh.

2. Now force a rescan by clicking the Rescan… link in the top right-hand corner – choose OK to the Rescan dialog box.

   Figure 3.4 shows the LUNs presented to ESX1 after the rescan has completed. If you select the vmhba40 software-device you should see the LUNs you presented in the IET configuration or the StarWind software. The IET iSCSI target displays separate LUNs (10 and 11) whereas the StarWind software displays each images device as separate SCSI targets (disk1 and disk2).

Figure 3.4



Configuring IET iSCSI Target with CHAP Authentication

So far, we have been connecting to the IET iSCSI target without any security. Security is implemented using CHAP Authentication. Very simply, a "user" is set up with a password which is required on the ESX server to allow access. It is only possible to have inbound (initiator to target authentication), as VMware does not currently support outbound (target to initiator authentication) with the software initiator.

Some systems require a password longer than 12 characters if authentication is used (this is defined under RFC 3720) and is enforced by some initiators, such as the Microsoft Initiator. However, it appears that VMware's Initiator doesn't, as I have used passwords shorter than 12 characters.

The only downside of enabling CHAP authentication for the first time is that you are required to reboot the ESX host.

1. On the IET server, nano –w /etc/ietd.conf
2. Un-remark the line:

IncomingUser

3. and replace with:

   IncomingUser *vmware 12charactersecret*

   Note:

   In this example, vmware is the user but this can be any name you like, followed by a secret which is greater than 12 characters. The secret is stored in the file as clear-text, so you must secure this file and login to the IET iSCSI server.

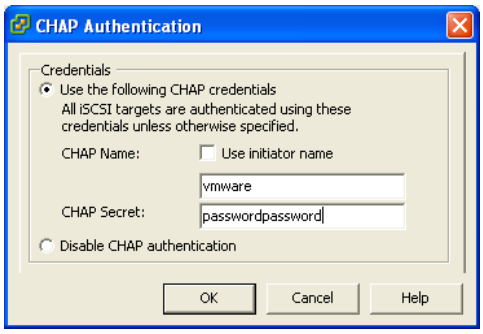
4. Restart the iSCSI Target daemon with:

   service iscsi-target start

5. Logon with the VI Client.
6. Select your ESX Host.
7. Click the Configuration Tab.
8. Choose in the Hardware pane, Choose Storage Adapters.
9. Under iSCSI Software Adapter select vmhba40.
10. Select Properties… and select the CHAP Authentication Tab.
11. Click Configure.
12. Choose Use the following CHAP credentials.
13. Click OK…. And choose OK to the reboot message.

In Figure 3.5 you can see that VMware does replace the CHAP Secret field with asterisks. This said the password is NOT stored in the dialog box as clear text. Confusingly, if you click "OK" to this dialog box and then return to it, the field is blank. This does not mean your change has not been recorded internally, just that it is simply not shown in the field.

Figure 3.5



Formatting Volumes with VMFS

VMFS is VMware's own proprietary file system. It now supports directories and a maximum of 3840 files in VMFS volume and 220 files in a directory. These numbers merely represent a "soft limit" rather than "hard limit." A VMFS volume can be used in virtual machines, templates, and ISO files. It has been designed to be safe with very large files such as virtual disks. Another key property is that VMFS supports multiple-access; more than one ESX host can presented the same LUNs formatted with VMFS without fear of corruption. This is achieved by using a sophisticated file and LUN locking system to prevent corruption of data, referred to as "SCSI Reservations."

To enhance performance and reliability, VMware improved VMFS in version 3 to significantly reduce the number and frequency of these reservations. Additionally, we now see these locks as dynamic. Whenever an ESX host powers on a VM, a file-level lock is placed on that VM's files. However, periodically the ESX host must go back to confirm it is still functioning, still running that VM – and still locking those files. If an ESX host fails to update these "dynamic locks," the files are forcibly unlocked. This is pretty critical in features like VMware HA. If locking was not dynamic, the locks would remain in place when an ESX host failed, and no other host would be able to power on that VM. That VM would be locked – and HA would not work.

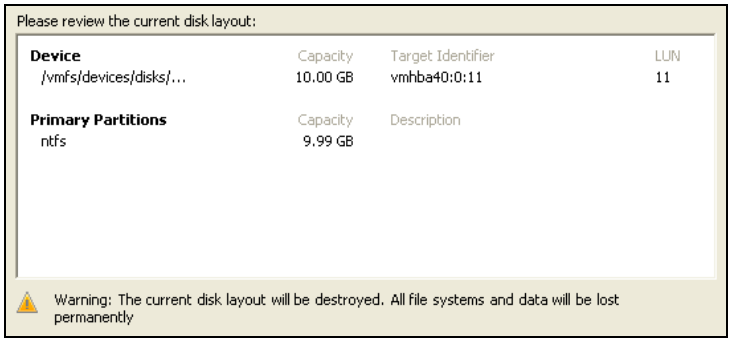
VMware does give guidelines for the number of ESX hosts that should be allowed access to a single VMFS volume. In ESX 2.x they stated no more than 10 ESX hosts should have access to a VMFS volume; in ESX 3.x this was increased to 32 ESX hosts.

VMFS uses a distributed journal. If a crash does occur, ESX no longer does a full "fsck" on VMFS volumes, but merely checks the journal. This is decidedly quicker – however, existing EXT3 partitions are still checked by the fsck method from Linux.

Formatting a VMFS volume is the same as if you are using local storage, SAN or iSCSI LUNs. As with any format procedure there is a huge potential for loosing data. When using the interface, take notice of any warnings that the Vi-Client produces as shown in Figure 3.6. There is no "undo format" facility in ESX 3.x, and so if you wipe terabytes of data during the process you would be resorting to your backup solution.

During the format process you will be asked to set a block size (1, 2, 4 and 8MB). These control the maximum file size which can be held in a given VMFS volume (256, 512, 1024, and 2048GB). Block sizes do greatly affect performance in VMFS and they also affect maximum file size. Sometimes this can be largely a theoretical issue, especially if your LUN size is less than the smallest file size 256GB. Generally, the larger the blocks size the better as this is good for performance and also allows you to have large VMDK files.

Figure 3.6



You will also receive the option to "maximize capacity" which sounds more complicated than it actually is. It merely means to use all of the LUN space for the VMFS volume. You can adjust this to make the VMFS volume smaller than the LUN size. There isn't any advantage to this and the free space left over is not easily accessible to the VI Client, anyway. While it is possible to create multiple VMFS volumes on a single LUN, this can affect performance. Although multiple ESX hosts can still access the VMFS volumes, doing so can impose a LUN wide SCSI reservation, which temporarily blocks access on multiple

VMFS volumes where only one VMFS volume might need locking. For these reasons I would recommend one VMFS to one LUN. In fact, I wish VMware would remove this option altogether from the VI Client.

Whenever possible, you should format VMFS volumes from the VI Client. One well-known way to optimize disk I/O is to keep the system from crossing track boundaries, known as "disk alignment." Master Boot Records frequently limits the number of hidden sectors to 63, which causes the default starting sector of disks that show more than 63 sectors per track to be sector 64. This can cause track misalignment, which defeats efforts to not cross track boundaries. The problem is further complicated by the characteristics of today's disks and controllers. Some disks don't accurately report track information to avoid other problems. Disks can also have a different number of sectors on the inner and outer tracks. "Disk alignment" is an effort to improve performance by keeping the number of sectors per second passing under the heads more or less constant. If you format volumes in the ESX installer or from the command-line using fdisk, the start sectors will not be 64k aligned. In contrast, the VI Client automatically ensures that disk alignment takes place.

GOTCHA:

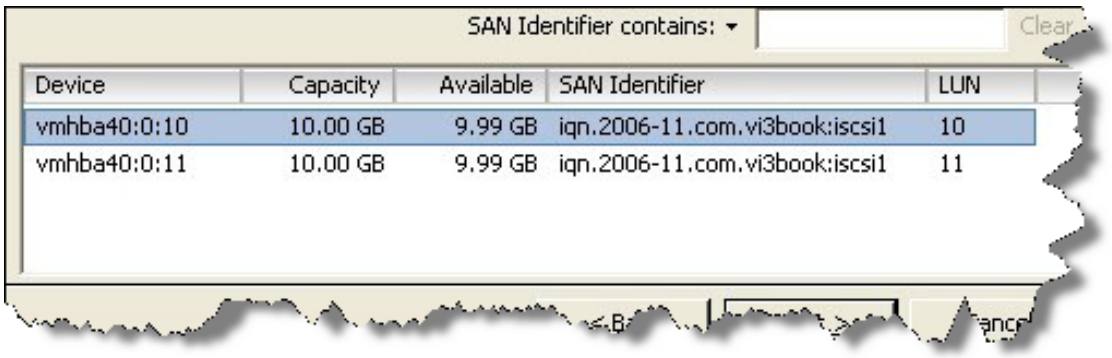
If you are using RDMs to access RAW or Native LUNs, disk alignment should be done within the guest operating system. This is because fundamentally it is Windows or Linux that is in charge of the file system. Tools like Microsoft's diskpart should be used if you think your VMs would benefit from disk alignment.

1. Select your ESX host.
2. Click the Configuration Tab.
3. In the Hardware Pane, choose Storage (SCSI, SAN, NFS).
4. Click the Add Storage… link.
5. Choose Disk/LUN.
6. Select the LUN you wish to format:

   Figure 3.7 shows the two iSCSI LUNs presented by the IET iSCSI target set-up in the iSCSI section of this chapter. Notice how the

SAN identifier, together with the iSCSI IQN, allows us to see clearly the type of storage and where it is located.

Figure 3.7



7. Set a datastore name such as iscsi1-lun10.
8. Set your maximum file size parameters.

   Note:

   As stated earlier its recommend to create one VMFS volume per LUN.

9. Click Finish.

Why don't I get all my space?

One common question asked is why, after formatting a 100GB LUN, you don't receive 100GB of VMFS storage. Like any file system, this storage system needs to have storage to function. In NTFS, this would be something like the Master File Table (MFT). In VMFS we refer to this as "metadata." The VMFS metadata is stored in files with the .SF extension in the root of the VMFS volume which "wastes" space on the disk. With a LUN or disk which is only 1.2GB in size, approximately 44% is wasted in metadata information. If you have a LUN of 10GB in size, about 4.8% is wasted in metadata information. If a LUN is 50GB in size, approximately 0.98% is wasted in metadata. So, the moral of the story is, the bigger the LUN, the smaller the percentage of metadata overhead.

Extending VMFS Volumes

Generally we create one VMFS partition per LUN, with the maximum size of a *single* VMFS volume being 2TB. The only way to exceed this limitation is by creating what VMware call "extents." Another reason to create an extent is if you are running out of space in a given VMFS volume.

The widespread use of SCSI-2 hardware causes 2TB's limitation, one that ESX shares with a number of x86 operating systems. SCSI-2 hardware uses a 16-byte address which then limits the maximum single LUN size. SCSI-3 equipment does not have this limitation, but we have not yet reached the tipping point for most OS vendors to move over to it completely. Fortunately, SCSI-3 equipment is backwards compatible with SCSI-2.

If you primarily use Windows, you will probably recognize extents. They function very similarly to Microsoft's "volume sets," or "spanned volumes." Most people in the VMware Community would recommend that you steer clear of this feature. I would agree with this assessment. I would only personally recommend extents as a workaround or band-aid.

In extents one LUN is stuck to another LUN to make the original VMFS larger. This could introduce performance headaches if the first LUN had a different RAID level or a different number of disk spindles. Any extents feature is only as good as the OS that underpins it – and most people would prefer a permanent "hardware" solution to this issue rather than a "software" workaround.

As great as the VMFS file system is, there are still some limitations. For example, there is no "Partition Magic" style tool for VMFS from VMware. This means that even if you have a SAN or iSCSI system that can dynamically make a LUN bigger from, say, 500GB to 1024GB, the original VMFS partition remains the original format size of 500GB. The only fix for this is either creating an extent or backup and restore. Or, if you have the free space, create a LUN that is 1024GB in size and copy the data in the 512GB LUN to the 1024GB. Afterwards, what you do with the empty VMFS and LUN is up to you. Most people would delete the smaller LUN so this capacity could be used elsewhere.

What this means from a storage perspective is that in your design you must carefully plan the LUN sizes – because changing them afterwards is difficult.

Another risky approach would be to use virtual storage and allocate more space to the ESX host than physically is available.

To extend an existing VMFS you need a free empty LUN. The VI Client makes a good job of hiding existing VMFS volumes to inhibit you from accidentally deleting valuable data. However, it does not hide any other LUNs with other OS file systems – so again be very careful, as there is no "undo extents" feature. To remove an extent, you remove the file system, along with any data.

The only real "gotcha" with extents is this scenario: let's say you later choose to extend a VMFS volume to make it bigger, and you add storage that makes the VMFS 2TB in size. It would still be formatted with the original block size. If this was 1MB you would still be limited to the maximum file size of 256GB caused by your original format.

With these caveats in mind, creating an extent is actually a very simple and easy process:

1. Select your ESX host.
2. Click the **Configuration** Tab.
3. In the **Hardware Pane** choose **Storage (SCSI, SAN, NFS).**
4. **Right-click the VMFS volume** you wish to make larger and choose **Properties.**
5. Click the **Add Extent** button.
6. Select a LUN from the list.

   Note:

   Before proceeding, confirm that the LUN is blank.
7. Click **Finish**

Configuring Multipathing with SAN

By virtue of having many HBAs in the ESX hosts and many SPs on the SAN array, it is possible to have many paths from one ESX host to a given LUN.

VMware does not currently use the multiple HBAs for load-balancing, merely for fault-tolerance. VMware takes the position that load-balancing to the storage is a more of a "storage vendors'" issue, and as a company, their expertise is in virtualization. Additionally, this would be difficult to encode from a VMkernel perspective, as they would need information about your SP's functionality to implement this. To do this, the VMkernel would have to be able to find out if a given SP was Active or Passive. An active SP can take a continual disk I/O load, whereas a passive SP only functions if the active fails. In this respect, an Active/Passive SAN would be unsuitable for load-balancing, whereas an Active/Active SAN would be a good candidate.

This aside, ESX does allow us to control which is the preferred path to a LUN on the SAN using the vmhbaA:T:L:V syntax. The first policy, which is the default, is called "Most Recently Used" (MRU) and is used with Active/Passive SPs on a SAN. The second policy is called "Fixed and Preferred" which is used with Active/Active SPs on a SAN. This second policy allows for manual disk I/O optimization.

MRU behaves in the following way: during installation, the ESX install scans the PCI bus looking for storage adapters. The first fiber-channel device found that the VMkernel has a driver for is set as the preferred path to the storage. The other paths from other SPs or HBAs are not used, except when this preferred path fails. When a failure occurs, a HBA connected to the passive SP takes over the disk I/0. The time taken to detect a failure and to use this redundant link is usually configured in the firmware of the HBA – it must be quick enough not to cause the VMkernel to think it has lost all connection to the storage. When the link on the active HBA returns (perhaps it was a failed fiber channel cable or switch) MRU does *not* return to the original HBA on the ESX host. In other words, MRU does fail-over, but not fail-back. If you wish to return to the original path, you must use the VI Client or command-line tools.

In contrast, the "Fixed and Preferred" path allows us to hard-code HBAs to service a given LUN. This enables us to configure that vmhba1 is the preferred path for LUN20 using SP0, whereas the preferred path for LUN21 is vmhba2 using SP1. This allows us to distribute the disk I/O across multiple HBAs and SPs. However, your SAN must support Active/Active on the SPs for this to work. When a failure occurs, the VMkernel is able to use any available path to the LUN, and when the link is restored it returns (Fixed) to original path. In other words, Fixed and Preferred does fail-over and fail-back.

Viewing Multiple Paths

You can view your multiple paths (if you have multiple HBAs) to a LUN within the Vi-client:

1. Select your ESX host.
2. Click the Configuration Tab.
3. Right-click a VMFS volume and choose Properties.

Figures 3.8 and 3.9 show the four paths available from two HBAs, each plugged into a different switch which is, in turn, plugged into two separate paths (vmhba1:0, 1:1, 2:0 and 2:1). We can see the vmhba1 is the active path (indicated in green) whereas the other 3 remaining paths are merely waiting for a failure to occur. Some storage people might disagree with this representation of the paths to the LUN. They may argue that if an ESX host has two adapters, it has two paths to a given LUN. Whatever your position is in this argument of 4 paths Vs 2 paths, precisely only one path is used at any one time. Incidentally, 32 is the maximum number of paths supported by VMware.

Figure 3.8

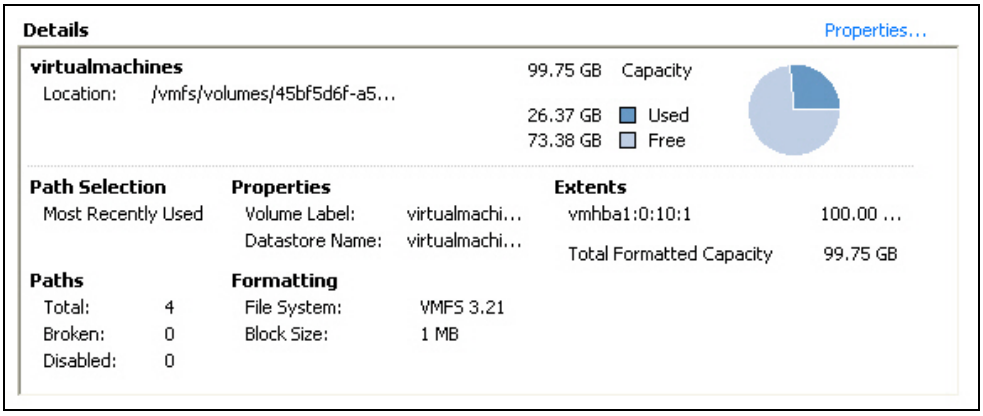
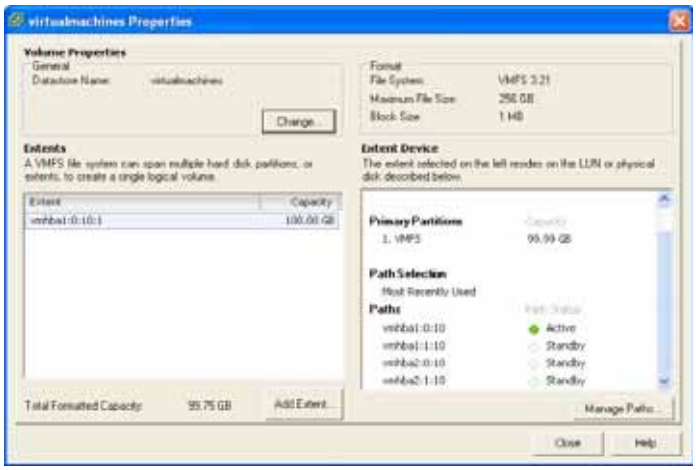


Figure 3.9



Note:

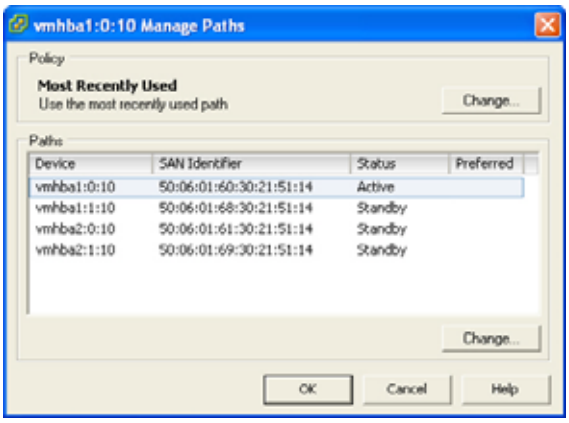
If a LUN is unformatted you can still see the paths by selecting the Storage Adapters link from the "Hardware Pane." Select your fiber-channel device and then right-click a path to a LUN (for example vmhba1:0:10)by choosing "Manage Paths."

Enabling Fixed and Preferred Paths

Warning: Only do this if you know you have a SAN which supports Active/Active storage controllers/processors!

1. Select your ESX host.
2. Click the Configuration Tab.
3. Right-click a VMFS volume, and choose Properties.
4. Click the Manage Paths button; figure 3.10 shows the current path status.

Figure 3.10

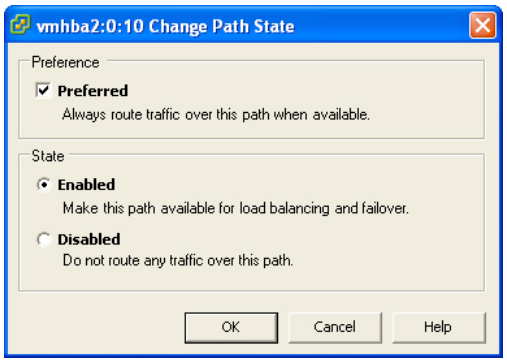


5. To Change the Policy Used, Click the Change button in the top right-hand corner of the dialog box, choose Fixed and click OK

To Change the Path Used

Click the path you wish to use, and click the Change button in the bottom right-hand corner of the dialog box. Set this path as being preferred – the disable option allows you stop a particular path from being used altogether. Figure 3.11 shows me making vmhba2 the preferred HBA for LUN 10 using SP0.

Figure 3.11



Figures 3.12 and 3.13 show the paths after changing the policy. Notice the asterisk (*) indicates which is the preferred path, and the status has changed to active for vmhba2:0:10.

Figure 3.12

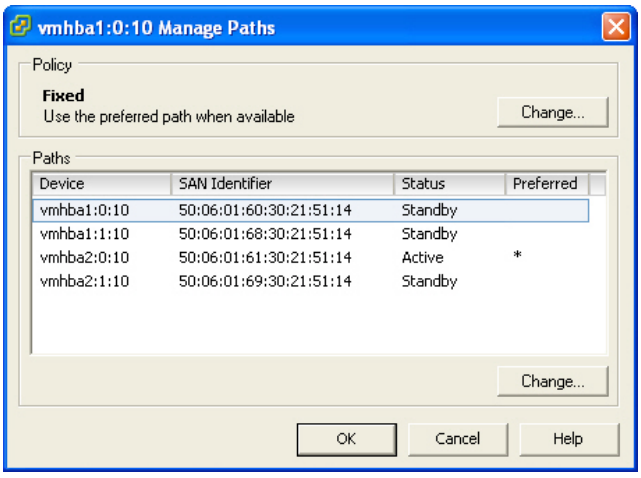
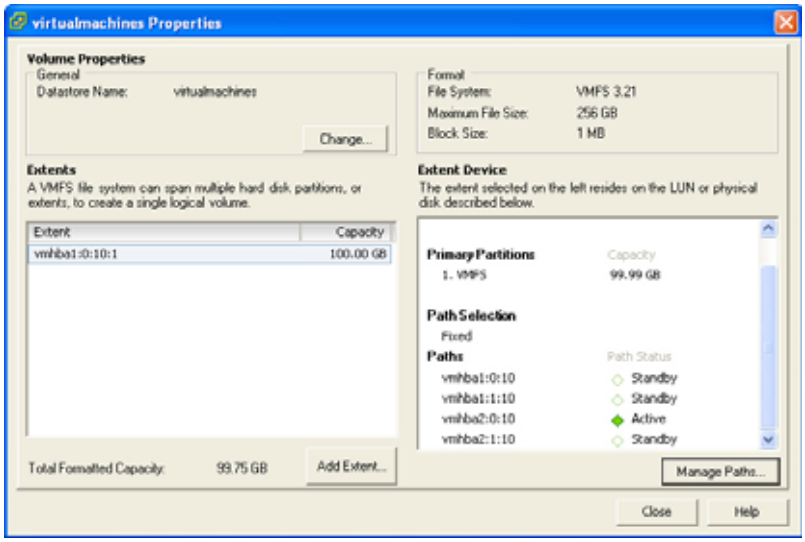


Figure 3.13



Broken Paths

Paths can and do fail, and the VI Client will alert you to this fact in the above dialog box. Figure 3.14 shows a dead path caused by a failed HBA, and Figure 3.15 shows a VMFS volume to which the ESX host has lost access.

Figure 3.14

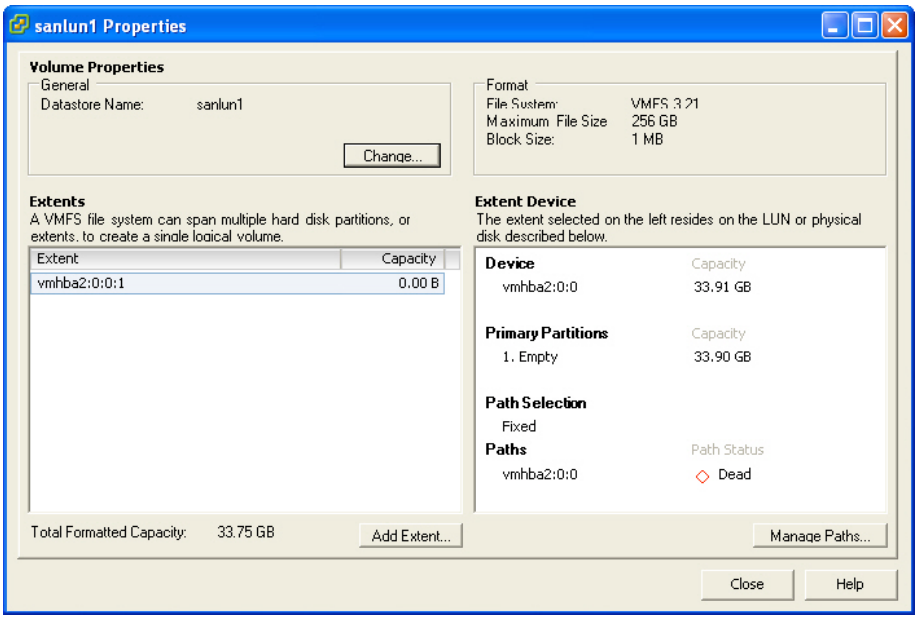


Figure 3.15

```
[root@esx1 root] # ls /vmfs/volumes/
478724-77d8fb-fc04-001560acef virtualmachines
```

Lastly, you can manage paths using the command-line tool called esxcfg-mpath. If you run esxcfg-mpath as root it will show you how to list paths, change path policies, and enable/disable paths.

Setting up NAS Storage

With the improvements and extension of the VMkernel's IP stack, VMware has introduced NAS support; more specifically Vmware supports the Linux "Network File System" (NFS) for file servers. This is where we have a file server which shares, or "exports," a folder.

Linux is supported natively by VMware using NFS and EXT3 as the files system. If you prefer to use Windows you will have to set up something like Microsoft's Services for UNIX (SFU) using NTFS partitions. Microsoft SFU emulates NFS support for a Windows Server. Fortunately, SFU is free – as long as you have paid for a valid Windows license. Alternatively, there are some very

good free virtual appliances such as OpenFile and FreeNAS which you could use instead. Whatever you decide to use as the platform for NFS it's important to know that VMware ESX server only supports version 3 for NFS using TCP as transport protocol.

VMware's NAS support has three main purposes

The main purpose is to make store ISOs of guest operating systems (Windows, Linux, Solaris, and Novell) rather than locating these on SAN or iSCSI based storage, which is the most expensive you can buy. If you have lots of ISOs they can soon take up a lot of space and you may prefer to put them on cheaper NAS storage.

We could also use NAS based storage to hold our templates. The down side is every time we went to create a new VM from a template this would generate a significant amount of network activity and would not be as fast or efficient as using SAN or iSCSI. This is especially true if you also factor in the odd occasional error. For example, if deploying a new template takes 40 minutes to complete, and then it fails for whatever reason at 99%, you would have to factor in another 40 minutes to try again.

Lastly, we could use NAS to hold running virtual machines. I suspect that VMware supports this mainly for test and development environments. The cool thing about the NAS support is that we don't need to spend our hard-earned dollars on a SAN or iSCSI just for test and development environments. NAS would be sufficient. Perhaps you're one of those guys with a truck load of hardware in your basement or garage and you want to set up your own Vi-3 environment but your budget won't stretch to even a MSA1000. If this is you, then NAS could be viable. Personally, I would prefer to use an iSCSI emulator because it would allow me to use VMFS.

It is also worth mentioning that there is a great deal you cannot do with NAS based storage. So I would doubt very much if anyone would seriously use NAS in a production environment except for ISO or template storage.

Here's a handy list of what's not supported by VMware with NAS:

- Boot ESX server
- VMFS
- Raw Device Mapping Files
- Clustering inside a virtual machine
- VMware Consolidated Backup

I will begin this section with a quick overview of the setup of "exports" with NFS using Linux. After that, I will demonstrate how to set Windows up with SFU. Lastly, I will show how to add in the exports to the ESX servers.

Using RedHat Linux

In this setup I used RHEL AS Release 3 (Taroon Update 2) as my NFS/NAS Server. I configured the RHEL machine with a 10GB boot disk and 50MB data disk with a folder I created called /iso.

1. Logon to your **NFS file server** and edit the **/etc/exports** file like so:

   ```
   /iso  172.168.3.0/24 (rw,no_root_squash,sync)
   ```

   Note:

   This allows the server, any ESX server with a VMkernel storage port group in the range of 172.168.0.0, to access the mounting point called /iso. RW enables this server read and write access. The default is that the root user does not get full access to the volume. The command "no_root_squash" allows applications like VirtualCenter read/write access to the volume. Normally, root access to NFS volumes is "squashed" (in other words, denied). You can specify additional security settings using /etc/hosts.allow and /etc/host.deny files associated with the portmap service/daemon. However, these are beyond the scope of this book.

2. Sync controls how data will be written back to the disks when services disconnect from the export or when the NAS is shutdown.

3. **Start your nfs service/daemon** with:

```
service nfs start
```

4. To make these NFS services start automatically at boot-up use the chkconfig utility which allows you to make safe changes to run-levels in Redhat Linux.

   ```
   chkconfig nfs on
   ```

Setting up NFS on Windows with SFU

These instructions are based on Windows 2003 with Service Pack 1. I have also successfully made SFU run on Windows 2000 with Service Pack 4.

Windows does not allow unchallenged access to shares without authentication. As Windows and the ESX Host do *not* share a common end-user database, we need some method of "mapping" the users on ESX Host to Windows. The method I have chosen is a simple mapping of the accounts using the files present on the ESX Host.

Installing SFU

1. **Copy the passwd** and **group files** from any one of your ESX servers; these are both held in **/etc**. You can use the free WinSCP tool to copy the files from the ESX host to your Windows Server.
2. **Extract the SFU package,** and run the **MSI package** called **SfuSetup.msi.**
3. Choose a **Custom Installation.**
4. **Expand Authentication tools for NFS,** and select the **User Mapping Service.**
5. Select **Next** to the **setupid** and **case-sensitive options dialog box.**
6. Under **Local User Name Mapping Service**, select **Password and Group files.**
7. **Type name and path** for passwd/group files, for example:

   c:\etc\passwd

c:\etc\group

8. Select the **Windows Domain.**
9. Select **Next**, **and accept the location for the install**…

   Note:

   Watch the status bar, check your email, make a cup of coffee, wonder how long you spend watching status bars… oh, and at the end of this - reboot your Windows/NFS Server.

Creating a User Mapping Between Administrator and Root

1. From the Start Menu, select Windows Services for UNIX.
2. Run the MMC, Services for UNIX Administration.
3. Select User Name Mapping node.
4. Choose Maps option, and under Advanced Maps, click Show User Maps.
5. Click List Windows Users button - and select Administrator.
6. Click List UNIX Users button - and Select root.
7. Click the Add button.

   Note:

   When the warning box appears – choose OK.
8. At the top of the console choose the Apply button.

Sharing out a folder

1. On the **Windows Explorer**, right-click a folder, and choose **Share and Security.**
2. Select the **NFS Sharing** tab.
3. Choose **Share this folder.**
4. Click the **Permissions** button, Select **X Allow root access**.
5. Change the **Type of Access to Read-Write.**

6. Choose **OK** to exit sharing dialogs.

   Note:

   Again if you're just using this share for ISOs you could leave this as read-only.

GOTHCA:

Watch out for CaseSensitivity on your sHaReNaMeS. Although Windows is not case-sensitive, it perfectly emulates NFS which *is case-sensitive*. If you want to remain sane, make them all lower-case with no spaces.

Confirming the Windows/NFS Server is functioning

Note: There are a number of tools we can use at the Windows/NFS server to see if things are working before adding in the NFS Share as IP Storage in the VI Client.

rpcinfo -p (lists listening ports on the server, notice TCP, NFS v3, Port 2049)

```
   program version protocol   port
-------------------------------------------------
    100000       2      udp         111    portmapper
    100000       2      tcp         111    portmapper
    351455       1      tcp         904    mapsVC
    100005       1      udp         1048   mountd
    100005       3      tcp         1048   mountd
    100021       1      udp         1047   nlockmgr
    100021       4      tcp         1047   nlockmgr
    100024       1      udp         1039   status
    100024       1      tcp         1039   status
    100003       2      udp         2049   nfs
    100003       3      udp         2049   nfs
    100003       2      tcp         2049   nfs
    100003       3      tcp         2049   nfs
```

showmount -e (exports list on nfs2.vi3book.com)

```
/iso                                    All Machines
```

ls -l

```
D:\sources\vmware\os-ISO's>ls –ls
total 11322628
1392128 -rwxrwxrwx+ 1 +Administrators  513   712769536
Feb  1  2005 wxp.iso
1251856 -rwxrwxrwx+ 1 +Administrators  513   640950272
Aug 25  2004 nt4.iso
```

Adding a NFS Mount Point

Adding a NFS mount point to an ESX host is the same whether you have used Linux NFS or Windows SFU. There is one option that does need explaining; as an ESX administrator you do have an over-ride option with permissions. Even if a NFS export has been set-up as Read/Write, you can mount it into ESX as read-only. This allows you to offer NAS as storage for ISOs but disallow it for other processes. Effectively, this would stop anyone from "accidentally" creating a VM on NAS storage when SAN or iSCSI would be preferred.

In my case, I used a FQDN name when specifying the NFS server. The Service Console is configured to resolve DNS names like this. If you have any doubts about name resolution try using nslookup at the Service Console on the name, and validate the results.

1. Next **Login** with the **VI Client.**
2. Choose the **ESX Host from the list**, and select **Configuration Tab** .
3. In the **Hardware pane**, select **Storage (SCSI, SAN, NFS).**
4. In the **right-hand side of the VI Client** click **Add Storage.**
5. In the Wizard, choose **Network File System.**
6. In the "**Locate Network File System**" page… complete the dialog box as follows:

   Server: Name of your NFS server, in my case **nfs1.vi3book.com**

Folder: Name of mount/volume you wish to access, in my case, **/iso**

DataStore Name: **nfs1-iso** (or anything you deem suitable)

GOTCHA:

Remember NFS comes from Linux – even if you're using Microsoft SFU export/share names ARE case-sensitive!

Occasionally, the NAS may be unavailable. If you want to force a reconnection to a NFS export you can do this through the Service Console with the esxcfg-nas command.

Note: If you want to check that you have communication to your IP storage, and test the VMkernel's IP stack, look at vmkping –D. Just like ping, it's a testing tool for the VMkernel.

Conclusion

In this chapter we examined the three main storage platforms available to ESX server. We also addressed how to format SAN and iSCSI LU's with VMFS. The last two chapters about networking and storage are pretty critical. Get your networking and storage sorted before you even begin creating V's and your life will be that much more trouble free. They are the absolute bedrock of the platform. Now that the ESX host server is correctly configured, we are now able to progress to the next chapter where we setup VirtualCenter.