Chapter 9

Modifying VMs

Solutions in this chapter:

- **The Virtual Machine VMDK File**
- **The Virtual Machine Configuration vmx File**
- **Converting IDE Drives to SCSI Drives**
- **Dynamic Creation of Virtual Machines**

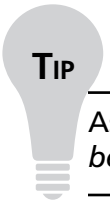
☑ **Summary**

Introduction

This chapter expands on the virtual machine's creation that was introduced in Chapter 21. To begin, we will discuss the two main components of a virtual machine, the .vmx and the .vmdk files. Then we will look at the hardware and version level of these files, as well as how we can change the files to be able to migrate a virtual machine's disk file from one VMware platform to another.

Virtual machines are made up of two files. The vmx file is the virtual machine's configuration file, while the virtual machine disk format (VMDK) file is the virtual machine's disk file or hard drive. We will examine these files and the different settings that can be used. Afterward, as an example, we will change a virtual machine's IDE disk to a SCSI disk.

To conclude, we will dynamically create a virtual machine using a script, as well as modify the script to build the virtual machine in a few different ways.



TIP

As a best practice, *always* make a backup of the files you are going to edit *before* you edit.

The Virtual Machine VMDK File

When working with virtual machines, there are two main components or files that need to be understood. The first is the VMDK file. But what exactly *is* the VMDK file? A virtual machine disk (VMDK) file is an encapsulation of an entire server or desktop environment in a single file. In a way, it can be seen as the hard drive for a virtual machine.

The VMDK file can have four different forms. Type 0 (monolithic sparse disk), Type 1 (growable; split into 2GB files), Type 2 (single pre-allocated; monolithic sparse disk), and Type 3 (pre-allocated; split into 2GB files). Types 1, 2, and 3 use a disk descriptor file, while type 0 does not. To make changes to the VMDK file, you need to be able to open and view the disk descriptor; otherwise, with the type 0 single disk, you would need to edit a very large binary file with a hex editor—an unwise choice. A better option, if you have the VMDK file on a VMFS file system, is to use vmkfstools to easily export the file in a Type 3 format.

For example:

```
Vmkfstools -e /mnt/bigspace/toputfile/thedisk.vmdk vmhba0:0:0:1:thedisk.dsk
```

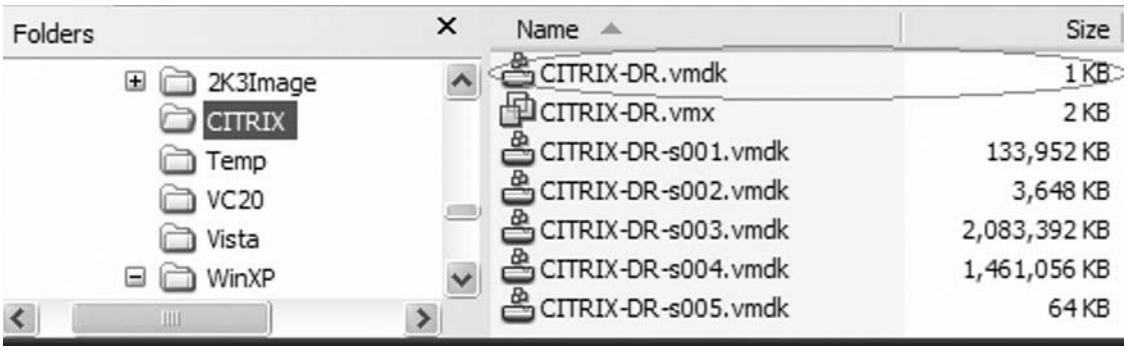
If you mount a file share to ESX and use the VMware File Manager to copy the VMDK file to this share, ESX uses the preceding command automatically when making the copy.

TIP

VMware does not support the use of VMDK files moved from a VMFS volume to a non-VMFS file system using SCP or FTP without first employing the vmkfstools export command or the file manager in the VMware Management Interface.

We should now have the VMDK file in a Type 1 growable split or a Type 3 preallocated split. You should now see a 1KB VMDK file. This is your disk descriptor file (see Figure 9.1).

Figure 9.1 The Disk Descriptor File



Using a text editor, we can open the disk descriptor file and view its contents. Code Listing 9.1 is one example of a disk descriptor file.

Code Listing 9.1 A Disk Descriptor File

```
# Disk DescriptorFile
version=1
CID=2af6d34d
parentCID=ffffffff
createType="twoGbMaxExtentSparse"
# Extent description
RW 4192256 SPARSE "Windows-s001.vmdk"
RW 4192256 SPARSE "Windows-s002.vmdk"
RW 4096 SPARSE "Windows-s003.vmdk"
# The Disk Data Base
#DDB
```

```
ddb.adapterType = "ide"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "16"
ddb.geometry.cylinders = "8322"
ddb.virtualHWVersion = "4"
ddb.toolsVersion =
```

VMDK Components

In the following subsections, we'll discuss the various parameters, settings, and commands related to VMDKs.

Version=1

The version parameter is the version of the disk descriptor file and not the VMDK file. Currently, in all VMware products, the disk descriptor version is 1.

```
# Disk DescriptorFile
version=1
```

CID=2af6d34d

Every time a VMware product opens up the vmx file, it creates a random 32-bit value and uses that value for the content identification or CID value.

parentCID=ffffffff

This parameter is the parent content identification which is used to specify whether the disk descriptor file is part of a snapshot file. If no snapshot file is being used, the value of this parameter is ffffffff.

file.createType="twoGbMaxExtentSparse"

The createType describes which type of file this is. There are currently 11 different values for this depending on the format of the data. Many values that exist in some products do not exist in others. The three values you see most often, especially with VMware's ESX server, are `"twoGbMaxExtentSparse"`, `"monolithicSparse"`, and `"monolithicFlat"`. Performing a manual change would make the disk unusable and has caused my VMware workstation host to crash. If you need to change the type of file, use the tool vmware-vdiskmanager to change the type.

```
# Extent description
RW 4192256 SPARSE "Windows-s001.vmdk"
RW 4192256 SPARSE "Windows-s002.vmdk"
RW 4096 SPARSE "Windows-s003.vmdk"
```

The preceding list shows files (typically VMDKs) that are used to store data blocks for the guest operating system. The values in those lines reveal the access mode of the VMDK,

the size in sectors of the VMDK, the type of the extent, and the location of the VMDK data file.

The Size in Sectors Value

The Size in Sectors value is required for a VMware Server to properly initialize the VMDK file. This value must be calculated based on the total byte size of the VMDK file and the number of bytes per sector. The Bytes per Sector is a static value of 512. The equation to calculate this value, as shown next, is quite simple.

Size in Sectors = (VMDK Byte Size – 512) / Bytes per Sector

The Disk Data Base Command

The Disk Data Base command will tell the virtual machine's hardware everything it needs to know to access the VMDK files. This is the actual disk geometry that the VMDK represents as a disk to the virtual machine. In Code Listing 9.2, this disk descriptor represents an IDE virtual disk with 63 sectors on 16 heads with 8,322 cylinders. It is important that the proper disk geometry be chosen to prevent "geometry mismatch" errors on the restored virtual machine (see Table 9.1).

Code Listing 9.2 A Disk Descriptor for an IDE Virtual Disk

```
# The Disk Data Base
#DDB
ddb.adapterType = "ide"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "16"
ddb.geometry.cylinders = "8322"
ddb.virtualHWVersion = "4"
ddb.toolsVersion = "6404"
```

Table 9.1 Disk Geometry

Disk Size	Heads	Sectors
<=1GB	64	32
>1GB and <=2GB	128	32
>2GB	255	63

Cylinders = (VMDK ByteSize – 512) / (Heads * Sectors * Bytes per Sector)
Three different adapter types can currently be used with virtual machines.

- **ide** For an IDE drive
- **buslogic** For a buslogic SCSI controller driver
- **lsilogic** For a lsilogic SCSI controller driver

One particular thing to notice in this section is the ddb.virtualHWVersion. This version number is the VMware platform the virtual machine is running on.

Swiss Army Knife…

Scripting the Backup of Virtual Machine's Configuration Files

In the next section, we will dig into the vmx configuration file for the virtual machines. Before that, however, let's put together a script to take care of one of the most important things we can do with these files: backing them up. This script is what I am using in my VMware ESX servers. They will back up all the configuration files for all the virtual machines, compress them into a tar file along with the vm-list file, and put them on a share on the network. The vm-list file is the list of registered virtual machines on an ESX server. This script runs daily and if I were to lose one of the ESX hosts, I could grab the backup file, register the virtual machines, and I am all set.

```
#!/bin/sh
# Virtual Machine VMX Backup
# Stephen Beaver
DOW=‘date +%a’ # Day of the week e.g. Mon
mount -t smbfs //server/share /mnt/smb -o
username=username/domain,password=password
SRC_DIR=/home/vmware/ #Directory will all vm configuration files
DST_DIR=/mnt/smb #Destination path which in this case is the mount point
BASE_DIR=/home #Base directory to put the vmlist file
HOST=“ESX-Server Name”
echo “src dir =”$SRC_DIR
echo “dst dir =”$DST_DIR
cp -f /etc/vmware/vm-list /home/vmware/vm-list
tar -czvf “$DST_DIR/vm_backup_$HOST-$DOW.tar.gz” “$SRC_DIR”
umount /mnt/smb
exit
```

The Virtual Machine Configuration vmx File

The vmx file is the configuration file that stores all the virtual machine's specific settings in one nice neat place. Code Listing 9.3 is an example of a vmx file.

Code Listing 9.3 A vmx File

```
#!/usr/bin/vmware
config.version = "6"
scsi0:0.present = "TRUE"
scsi0:0.name = "ESX_SAN4:2K900.vmdk"
scsi0:0.mode = "persistent"
scsi0.present = "true"
scsi0.virtualDev = "vmxbuslogic"
memSize = "512"
displayName = "2K900"
guestOS = "win2000Serv"
ethernet0.present = "true"
ethernet0.connectionType = "monitor_dev"
ethernet0.devName = "bond0"
ethernet0.networkName = "FH_Network"
Ethernet0.addressType = "vpx"
Ethernet0.generatedAddress = "00:50:56:9d:4d:10"
Ethernet0.virtualDev = "vmxnet"
floppy0.present = "true"
floppy0.startConnected = "false"
ide1:0.present = "true"
ide1:0.fileName = "/dev/cdrom"
ide1:0.deviceType = "atapi-cdrom"
ide1:0.startConnected = "FALSE"
draw = "gdi"
uuid.bios = "50 1d 07 5c a9 f3 2b dd-8b 3e 83 10 b2 ea 89 0b"
uuid.location = "56 4d b5 45 28 5a b0 20-29 52 da f8 22 74 60 1d"
uuid.action = "keep"
priority.grabbed = "normal"
priority.ungrabbed = "normal"
isolation.tools.dnd.disable = "TRUE"
suspend.Directory = "/vmfs/vmhba1:0:83:1"
autostart = "true"
autostop = "softpoweroff"
tools.syncTime = "FALSE"
```

This vmx file came from one of my virtual machines on a VMware ESX server. Let's take a look at the different settings in the file. As a rule, virtual machines will only read the full vmx file when the virtual machine is powered on. Thus, you should edit the virtual machine's vmx file when the virtual machine is off only. I have come across this scenario while playing around in the lab. There, I had a virtual machine and made a manual change to the configuration file. ESX knew I made a change and so it paused the virtual machine to ask me a question: "The configuration file for this VM has changed. Do you wish to reload the configuration file?" If the virtual machine in my production environment had instead been paused, I would have had a few people to answer to.

```
config.version = "6"
scsi0:0.present = "TRUE"
scsi0:0.name = "ESX_SAN4:2K900.vmdk"
scsi0:0.mode = "persistent"
scsi0.present = "true"
scsi0.virtualDev = "vmxbuslogic"
```

vmx File Components

In this subsection, we'll discuss the various parameters, settings, and commands related to vmx files.

config.version = ""

This is the hardware version level. When we talked about downgrading the disk descriptor file, this is what we must change to control the hardware version so it will work in the different products. What we see next are the settings for the SCSI drive. Scsi0:0 is the virtual machine's boot drive.

Scsi0:0.present = ""

This lets the host know that the virtual machine has a SCSI drive present. This can have an entry of True or False.

Scsi0:0.name = ""

This is the name and path of the VMDK file that the virtual machine will use. In the earlier example, "ESX_SAN4:2K900.vmdk" points to a common name of a LUN on the SAN called ESX_SAN4, and the 2K900.vmdk is the disk file located on the LUN.

Scsi0:0.mode = ""

This setting is the mode of the disk file. The following four disk modes are available.

- **Persistent** Changes are immediately and permanently written to the virtual disk.
- **Nonpersistent** Changes are discarded when the virtual machine powers off.
- **Undoable** Changes are saved, discarded or appended at your discretion.
- **Append** Changes are appended to a redo log when the virtual machine powers off.

scsi0.present = “”

This setting lets the host know this virtual machine has a SCSI controller. The value can be True for present, and False for no SCSI.

scsi0.virtualDev = “”

This setting determines what SCSI drivers the controller is using. Two different values can be used here.

- **vmxbuslogic** When using the buslogic SCSI driver
- **vmxlsilogic** When using the lsilogic SCSI driver

These are also the settings we would change on the vmx file to switch from an IDE disk to a SCSI.

The next part of the configuration vmx file is the memory, name, and guestOS, all of which do not need much explanation:

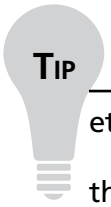
- **memSize = “512”** How much memory the virtual machine is allocated
- **displayName = “2K900”** The display name of the virtual machine
- **guestOS = “win2000Serv”** Which operating system the VM is running

The next part concerns the Ethernet adapter and whether Virtual Center is used to monitor this virtual machine (see the following example).

```
ethernet0.present = "true"
ethernet0.connectionType = "monitor_dev"
ethernet0.devName = "bond0"
ethernet0.networkName = "FH_Network"
Ethernet0.addressType = "vpx"
Ethernet0.generatedAddress = "00:50:56:9d:4d:10"
Ethernet0.virtualDev = "vmxnet"
```

ethernet0.present = “”

This value defines whether the network settings are read and processed. This value can be “true” or “false.” If the value is “true,” then all other parameters are then processed. If the value is “false,” then all other network parameters for that device are ignored.



TIP

ethernet0.startConnected = "true"

Ethernet0.present = "true" also sets startConnected to "TRUE", though this may not appear in the vmx (another silent default).

So if you want the device to be present—but not at boot-time—you must use ethernet0.startConnected = "FALSE".

ethernet0.connectionType = ""

This parameter concerns virtual networks. Your choices for this value are "bridged", "hostonly", "nat", "monitor_dev", and "custom". The custom settings are an expert way to use a combination of "connectionType" and "vne.t". A good example of this would be the following:

```
ethernet0.connectionType = "CUSTOM"
```

And the exact number of the VNET you want might look like:

```
ethernet0.vnet = "VMNET0"
```

ethernet0.devName = ""

This parameter is the actual name of the device being used. This could be one of the virtual ethernet cards like vmnic0, or in this case a bond of two ethernet cards together called "bond0".

ethernet0.networkName = ""

This is the name of the virtual switch that the virtual machine will be using for networking. In this example, the virtual switch's name is FH_Network.

Ethernet0.addressType = "vpx"

This parameter is only present when the virtual machine is on an ESX server that is controlled by Virtual Center.

Ethernet0.generatedAddress = ""

This parameter is the MAC address of the virtual machine. In this case, the MAC address is generated by the host application.

VMware has a special range of MAC addresses that are allocated for the virtual machines. The following lists the different ranges of addresses.

- **00:05:69:00:00:00** Automatically assigned by MUI when building a VM without VirtualCenter (ESX <2.0)
- **00:0c:29:00:00:00** Automatically assigned by MUI when building a VM without VirtualCenter as well as the other VMware products (ESX 2.0 +, all VMware)

- **00:50:56:00:00:00 – 00:50:56:3f:ff:ff** Manually configured MACs
- **00:50:56:80:00:00 – 00:50:56:bf:ff:ff** VirtualCenter-generated MACs

Ethernet0.virtualDev = "vlance" or "vmxnet" or "e1000"

This parameter is to define the virtual adapter itself. The choices available are

- **vlance** This is based on the AMD PCNet 32 and has the most backward compatibility. Take note that if you use vlance with your virtual machine, the VM will only show what it is connected at 10mb. This is presented for backward compatibility only and does not represent the actual speed with which the VM is communicating. The VM will use all the bandwidth given to it.
- **vmxnet** This is a VMware custom high-performance vmxnet virtual network adapter which allows for faster networking performance. This is the adapter you should use whenever possible, given it offers better performance than the vlance driver and less overhead.
- **e1000** This is the Intel pro 1000 adapter, which is the default virtual NIC when choosing a 64-bit guest. It can be manually edited in the config file.

Floppy Drives and CD-ROMs for Virtual Machines

The following parameter is the configuration of the floppy and CD-ROM for the virtual machine. Notice that I have startConnected set to "false" for these devices. As a rule of thumb, I recommend leaving these disconnected until you need them.

```
floppy0.present = "true"
floppy0.startConnected = "false"
ide1:0.present = "true"
ide1:0.fileName = "/dev/cdrom"
ide1:0.deviceType = "atapi-cdrom"
ide1:0.startConnected = "false"
```

Notice that the parameter ide1:0.fileName is currently set to "dev/cdrom." This is the emulation of the CD-ROM device that shows up as a VMware CD-ROM and not the actual physical host CD-ROM device. By changing the fileName and deviceType values, you can also mount ISO images to the virtual machine.

```
ide1:0.fileName = "/iso/nameof.iso
ide1:0.deviceType = "cdrom-image"
```

Graphics Emulation, Unique Identifiers

VMware products offer two modes for host emulation of the graphics inside the virtual machine: GDI (Graphics Device Interface; the classic Windows graphics mode) and

DirectDraw (a mode designed for games and other applications that write directly to the hardware).

```
draw = "gdi"
```

In general, Windows guest operating systems (Windows 95, Windows 98, Windows NT, and Windows 2000) perform better in GDI mode than in DirectDraw mode, while Linux guest operating systems (or any guest operating systems that use an X server) run much better in DirectDraw mode.

WARNING

DirectDraw on Windows 2000 is fairly buggy, so the virtual machine displays a cautionary message if you try to enable it. In addition, some specific issues have been identified on both Windows NT and Windows 2000 hosts when the virtual machine is using DirectDraw mode.

Once you start a virtual machine, the VMware host will then generate another two lines to identify the virtual machine. Whenever you change the path to the vmx-file, either by renaming or moving to a different location, VMware wants to update these lines to reflect that change (see the following example).

```
uuid.location = "56 4d ee 3c 52 06 a3 de-be 4a 73 9c cc 99 15 1f"
uuid.bios = "56 4d ee 3c 52 06 a3 de-be 4a 73 9c cc 99 15 1f"
```

If you've ever moved a virtual machine from one host to another, then when you start the machine you've probably seen a message similar to this:

> The virtual machine's configuration file has changed its location since its last poweron. Do you want to create a new unique identifier (UUID) for the virtual machine or keep the old one?

Your choices are Keep, Create, Always Keep, and Always Create. If you choose **Always Keep** or **Always Create**, then the parameter uuid.action is added to the vmx file (see the following example).

```
uuid.action = "Keep" or "Create"
```

The values you can use here are Keep or Create for Always Keep and Always Create.

Priority, VMware Tools Settings, and Suspend

The "grabbed: HIGH - ungrabbed: NORMAL" setting is useful if you have many background processes or applications and you do not care if they run with fairly low relative priority while a virtual machine is in the foreground. In return, you get a very noticeable performance boost using a virtual machine while another virtual machine is running or

while some other processor-intensive task (a compile, for example) is running in the background (see the following example).

```
priority.grabbed = "high" or "normal"
```

The reverse is true of the "grabbed: NORMAL - ungrabbed: LOW" setting. If your host machine feels too sluggish when a virtual machine is running in the background, you can direct the virtual machine to drop its priority when it does not have control of the mouse and keyboard. As with the high setting, this is a heavy-handed change of priority, so the virtual machine (and any background applications inside) runs much more slowly.

```
priority.ungrabbed = "normal" or "low"
```

isolation.tools.dnd.disable = "True" or "False"

This setting is to enable/disable Host/Guest drag and drop interface. The values you can use here are "True" and "False".

suspend.Directory = "/vmfs/vmhba1:0:83:1"

This parameter is the location the host should use to "suspend" a virtual machine. The following example was taken from an ESX server that is attached to a SAN. Notice that the path is made up of the true path vmhba1:0:83:1 and not the friendly name that I set for the LUN: /vmfs/ESX_SAN4/.

Autostart, Autostop, and Time Sync Options

In this section, we'll discuss autostart, autostop, and time sync options that you can be used for configuring a virtual machine. The following example shows autostart and autostop command scripts.

```
autostart = "true" or "false"
autostop = "softpoweroff" or "poweroff"
autostart.order = ""
autostop.order = ""
```

You can configure a virtual machine to automatically begin when the host starts up from a reboot and also to automatically power off or shut down the guest OS when the host is being shut down. When you utilize this option, the autostart and autostop options are added to the virtual machine's vmx file. You can also take this a step further and define the startup and shutdown order of the virtual machines using the autostop.order and autostart. order. By default, it would use order number x10. To give you an example, if you wanted VM1 to be the first virtual machine started and the last virtual machine to shutdown, you would set the configuration this way:

```
autostart.order = "10"
autostop.order = "10"
```

To change this to be the third virtual machine started, change the number from 10 to 30.

```
tools.syncTime = "FALSE" or "TRUE"
```

The tools.syncTime Option

The last option in my vmx file is the tools.syncTime. This option is used to determine if the virtual machine is going to update its time with the host time via the VMware tools or not.

Virtual Machine Conversion from IDE to SCSI

You may find the need to be able to move virtual machines around from one platform to another. For example, I encourage people to utilize VMware Workstation in order to work on a virtual machine while on the go. I have had several instances where a virtual machine was created on VMware Workstation, but unfortunately was not created in legacy mode or had an IDE drive. As a result, when attempting to migrate to ESX, it would fail until some changes were made.

Therefore, here we will examine changing an IDE drive to a SCSI drive. Before we change the settings, we need to get the SCSI drivers in the system first. The easiest way to do this is to add another hard disk to the virtual machine as a secondary drive. Configure this drive to be a SCSI drive. Start the virtual machine with the new drive attached and, the SCSI drivers are now in place, allowing us to continue and really edit the files. When we open the descriptor file for a virtual machine using an IDE drive, it looks like the sample in Code Listing 9.4.

Code Listing 9.4 Descriptor File for a Virtual Machine Using an IDE Drive

```
# Disk DescriptorFile
version=1
CID=2af6d34d
parentCID=ffffffff
createType="twoGbMaxExtentSparse"
# Extent description
RW 4192256 SPARSE "Windows-s001.vmdk"
RW 4192256 SPARSE "Windows-s002.vmdk"
RW 4096 SPARSE "Windows-s003.vmdk"
# The Disk Data Base
#DDB
ddb.adapterType = "ide"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "16"
ddb.geometry.cylinders = "8322"
ddb.virtualHWVersion = "4"
ddb.toolsVersion = "6404"
```

Starting with the ddb.adapterType you can see that this was indeed an IDE drive. There are a total of three different options for this setting. We'll discuss each in this section.

ddb.adapterType = "buslogic"

This entry converts the disk into a SCSI-disk with a BusLogic Controller. This is the standard for Windows 2000 virtual machines.

ddb.adapterType = "lsilogic"

This entry converts the disk into a SCSI-disk with LSILogic Controller. This is the standard for Windows 2003 virtual machines.

```
ddb.adapterType = "ide"
```

This entry converts the disk into an IDE-disk with Intel-IDE Controller.

Next, let's open the SCSI disk that we used to get the drivers in the virtual machine and use it to give us the section, heads, and cylinder values we need.

```
ddb.adapterType = "buslogic"
ddb.geometry.cylinders = "522"
ddb.geometry.heads = "255"
ddb.geometry.sectors = "63"
```

Put this all together and we have a new SCSI disk for our virtual machine.

There is one change left to be done, however. We will need to change the ddb. virtualHWVersion. The ddb.virtualHWVersion is dependent upon which VMware platform you are using. You may need to change the version number to get the virtual machine to start in certain cases, namely moving a virtual machine in to ESX Server.

Change the ddb.virtualHWVersion = "4" and make it ddb.virtualHWVersion = "3". You now have a legacy virtual machine disk file you have converted from IDE to SCSI. You've also brought the virtual machine disk file down to legacy mode so that it can run on ESX.

```
# Disk DescriptorFile
version=1
CID=826d3b6e
parentCID=ffffffff
createType="twoGbMaxExtentSparse"
# Extent description
RW 4192256 SPARSE "Windows-s001.vmdk"
RW 4192256 SPARSE "Windows-s002.vmdk"
RW 4096 SPARSE "Windows-s003.vmdk"
# The Disk Data Base
#DDB
```

```
ddb.adapterType = "buslogic"
ddb.geometry.sectors = "63"
ddb.geometry.heads = "255"
ddb.geometry.cylinders = "522"
ddb.virtualHWVersion = "3"
ddb.toolsVersion = "6309"
```

To complete this process we need to make an adjustment in the vmx file in order to change the IDE values to SCSI. Code Listing 9.5 is an example of a disk file that's been configured to use an IDE.

Code Listing 9.5 Configuring a Disk to Use an IDE

```
config.version = "8"
virtualHW.version = "4"
scsi0.present = "TRUE"
memsize = "200"
ide0:0.present = "TRUE"
ide0:0.fileName = "Windows.vmdk"
ide1:0.present = "TRUE"
ide1:0.fileName = "auto detect"
ide1:0.deviceType = "cdrom-raw"
floppy0.fileName = "A:"
ethernet0.present = "TRUE"
usb.present = "TRUE"
sound.present = "TRUE"
sound.virtualDev = "es1371"
displayName = "Windows XP Professional 1"
guestOS = "winxppro"
nvram = "winxppro.nvram"
ide0:0.redo = ""
ethernet0.addressType = "generated"
uuid.location = "56 4d b7 df d7 1d 42 ca-3e 81 5d a3 5e 05 7a f7"
uuid.bios = "56 4d b7 df d7 1d 42 ca-3e 81 5d a3 5e 05 7a f7"
tools.remindInstall = "FALSE"
ethernet0.generatedAddress = "00:0c:29:05:7a:f7"
ethernet0.generatedAddressOffset = "0"
ide1:0.autodetect = "TRUE"
ide1:0.startConnected = "TRUE"
tools.syncTime = "FALSE"
```

To finish the change from IDE to SCSI we need to adjust these lines in the vmx file (see Table 9.2).

Table 9.2 VMX Old and New Settings

From the Old Settings	To the New Settings
config.version = "8"	config.version = "6"
virtualHW.version = "4"	virtualHW.version = "3"
ide0:0.present = "TRUE"	scsi0.present = "TRUE"
ide0:0.fileName = "Windows.vmdk"	scsi0:0.present = "TRUE"
	scsi0:0.fileName = "Windows.vmdk"

Now we have completed downgrading the virtual hardware and also changed a virtual machine from using an IDE drive to a SCSI drive. This virtual machine will now start and run in VMWare ESX server. By using the example of taking a virtual machine from VMware Workstation and getting it to run to VMware ESX Server, we have gone from one extreme of the VMware product line (workstation) to the other extreme (ESX Server).

Scripted Disconnect of IDE Devices

As a general rule, you should always have the CD-ROM and floppy drive disconnect so they don't take away resources from the service console. This is also true if you place a CD-ROM in the physical host's drive, because all the virtual machines will not start to autorun the CD-ROM. VMotion also won't work if either the CD-ROM or the floppy is connected. The script shown in Code Listing 9.6 will disconnect all these devices in virtual machines that are registered on ESX Server. This script was originally posted on the VMware community forum by Stuart Thompson (aka, Mr-T) and Matt Pound, and it includes a few additions by me.

Code Listing 9.6 Disconnecting Devices in Virtual Machines Registered on an ESX Server

```
#!/bin/bash
# IDE / Floppy Disconnect Script
# Script by: Stuart Thompson and Matt Pound
# Edit by: Steve Beaver (Added floppy drive)
vmwarelist='vmware-cmd -l'
vmwarelist='echo $vmwarelist | sed -e 's/ /*/g''
vmwarelist='echo $vmwarelist | sed -e 's/.vmx/.vmx /g''
for vm in $vmwarelist
do
  vm='echo $vm | sed -e 's/*/ /g''
  vm='echo $vm | sed -e 's/ \//*/g''
```

```
if [ 'vmware-cmd "$vm" getstate | sed -e 's/getstate() = //'' = "on" ]
then
echo Looking @ $vm
IDEBUS='seq 0 1'
for i in $IDEBUS;
do
  echo BUS : $i
    IDEDEVICE='seq 0 1'
  for j in $IDEDEVICE;
  do
      PRESENT='vmware-cmd "$vm" getconfig ide$i:$j.present | cut -f3 -d " "'
      if [ $PRESENT = "true" ]
      then
        TYPE='vmware-cmd "$vm" getconfig ide$i:$j.deviceType | cut -f3 -d " "'
          if [[ $TYPE == "atapi-cdrom" || $TYPE == "cdrom-image" ]]
        then
          echo Found CDROM on IDE$i:$j
          vmware-cmd "$vm" disconnectdevice ide$i:$j
        fi
      fi
  done
done
fi
done
```

Swiss Army Knife…

vmwarelist='vmware-cmd -l'

You can change this value to point to a specific path of a virtual machine and have these scripts set up to run on only one virtual machine instead of all virtual machines.

Vmwarelist='/home/vmware/vmserver/vmserver.vmx

Employing this script as a base, you can choose many options using the vmware-cmd to make a change to all of your registered virtual machines. Take a look at Code Listing 9.7, which shows how you can start all your registered machines.

Code Listing 9.7 Starting All Registered Virtual Machines

```
#!/bin/bash
vmwarelist='vmware-cmd -l'
vmwarelist='echo $vmwarelist | sed -e 's/ /*/g''
vmwarelist='echo $vmwarelist | sed -e 's/.vmx/.vmx /g''
for vm in $vmwarelist
do
  vm='echo $vm | sed -e 's/*/ /g''
  vm='echo $vm | sed -e 's/ \//*/g''
  if [ 'vmware-cmd "$vm" getstate | sed -e 's/getstate() = //'' = "off" ]
  then
    echo Found $vm that is off, Starting $vm
    vmware-cmd "$vm" start
  fi
done
```

Now, let's take a look at a script to stop those virtual machines that are running.

```
#!/bin/bash
vmwarelist='vmware-cmd -l'
vmwarelist='echo $vmwarelist | sed -e 's/ /*/g''
vmwarelist='echo $vmwarelist | sed -e 's/.vmx/.vmx /g''
for vm in $vmwarelist
do
  vm='echo $vm | sed -e 's/*/ /g''
  vm='echo $vm | sed -e 's/ \//*/g''
  if [ 'vmware-cmd "$vm" getstate | sed -e 's/getstate() = //'' = "on" ]
  then
    echo Found $vm that is on, Stopping $vm
    vmware-cmd "$vm" stop trysoft
  fi
done
```

Code Listing 9.8 is one more example of this script, which will reboot all of the running virtual machines. This is very handy if you have installed updates or anything else and want to delay the reboot till later.

Code Listing 9.8 Script for Rebooting All Running Virtual Machines

```
#!/bin/bash
vmwarelist='vmware-cmd -l'
vmwarelist='echo $vmwarelist | sed -e 's/ /*/g''
vmwarelist='echo $vmwarelist | sed -e 's/.vmx/.vmx /g''
```

```
for vm in $vmwarelist
do
  vm='echo $vm | sed -e 's/*/ /g''
  vm='echo $vm | sed -e 's/ \//*/g''
  if [ 'vmware-cmd "$vm" getstate | sed -e 's/getstate() = //'' = "on" ]
  then
    echo Found $vm that is on, Rebooting $vm
    vmware-cmd "$vm" reset trysoft
  fi
done
```

Dynamic Creation of Virtual Machines

Now that we have looked at what makes up the vmx file, let's generate some scripts to dynamically create virtual machines. First, we'll take a script and modify it so we can create a virtual machine that will use a golden image as its base. We'll then make a couple of changes so we can take advantage of Altiris in the VM creation. We will then modify the script so that a virtual machine will be created and then start the VM with the installation CD mounted to begin the installation.

Code Listing 9.9 shows script that uses a golden image disk file. A golden image disk file is a fully loaded and patched virtual machine vmx file that has had sysprep run on it so it can be cloned.

WARNING

Please make sure you look through these scripts and make any changes needed to match your environment. Pay attention to the vmhba path and double-check these values with the values in your own environment.

Code Listing 9.9 Using a Golden Image Disk File to Dynamically Create a Virtual Machine

```
#!/bin/bash
#Scripting VMware Power Tools: Automating Virtual Infrastructure Administration
#Dynamic Creation of a new Virtual Machine using a Golden Image
#Stephen Beaver
#####USER MODIFICATION#################
#VMNAME is the name of the new virtual machine
#VMOS specifies which Operating System the virtual machine will have
#GLDIMAGE is the path to the "Golden Image" VMDK file
```

```
#DESTVMFS is the path to VMFS partition that the VMDK file
####################################
VMOS="winNetStandard"
VMMEMSIZE="256"
GLDIMAGE="/vmfs/FHVMFS1/Windows_2003_Standard.vmdk"
DESTVMFS="vmhba0:0:0:10"
#####END MODIFICATION#####
LOG="/var/log/$1.log"
echo "Start of Logging" > $LOG
echo "Importing Golden Image Disk File VMDK" >> $LOG
vmkfstools -i $GLDIMAGE $DESTVMFS:$1.vmdk
echo "Creating VMX Configuration File" >> $LOG
mkdir /home/vmware/$1
exec 6>&1
exec 1>/home/vmware/$1/$1.vmx
# write the configuration file
echo #!/usr/bin/vmware
echo config.version = '"'6'"'
echo virtualHW.version = '"'3'"'
echo memsize = '"'$VMMEMSIZE'"'
echo floppy0.present = '"'TRUE'"'
echo usb.present = '"'FALSE'"'
echo displayName = '"'$1'"'
echo guestOS = '"'$VMOS'"'
echo suspend.Directory = '"'/vmfs/vmhba0:0:0:10/'"'
echo checkpoint.cptConfigName = '"'$1'"'
echo priority.grabbed = '"'normal'"'
echo priority.ungrabbed = '"'normal'"'
echo ide1:0.present = '"'TRUE'"'
echo ide1:0.fileName = '"'auto detect'"'
echo ide1:0.deviceType = '"'cdrom-raw'"'
echo ide1:0.startConnected = '"'FALSE'"'
echo floppy0.startConnected = '"'FALSE'"'
echo floppy0.fileName = '"'/dev/fd0'"'
echo Ethernet0.present = '"'TRUE'"'
echo Ethernet0.connectionType = '"'monitor_dev'"'
echo Ethernet0.networkName = '"'Network0'"'
echo draw = '"'gdi'"'
echo
echo scsi0.present = '"'TRUE'"'
```

```
echo scsi0:1.present = '"'TRUE'"'
echo scsi0:1.name = '"'$DESTVMFS:$1.vmdk'"'
echo scsi0:1.writeThrough = '"'TRUE'"'
echo scsi0.virtualDev = '"'vmxlsilogic'"'
echo
# close file
exec 1>&-
# make stdout a copy of FD 6 (reset stdout), and close FD6
exec 1>&6
exec 6>&-
echo "VMX Configuration File Created Successfully" >> $LOG
#Change the file permissions
chmod 755 /home/vmware/$1/$1.vmx
#Register the new VM
echo "Registering .vmx Configuration" >> $LOG
vmware-cmd -s register /home/vmware/$1/$1.vmx
echo "VMX Initialization Completed Successfully" >> $LOG
```

NOTE

Notice that the preceding script uses a golden image file that is local to that machine. If your golden image is located on a network share, you can easily mount that share and import the file from there. To mount a network share you can use the following command:

mount-t smbfs //server/share /mnt/smb -o username=username/domain, password=password

Next, we'll take the same script and make a few changes so it will work with an ESX Server managed with Altiris. At the end of this script, the virtual machine is started and should boot PXE, which Altiris can then take over and use to install the operating system (see Code Listing 9.10).

Code Listing 9.10 Creating a New Virtual Machine to Use with an ESX Server Managed by Altiris

```
#!/bin/bash
#Scripting VMware Power Tools: Automating Virtual Infrastructure Administration
#Creates a new Virtual Machine for use with Altiris
#Stephen Beaver
#####USER MODIFICATION################
```

```
#VMNAME is the name of the new virtual machine
#VMOS specifies which Operating System the virtual machine will have
#DESTVMFS is the path to the VMFS partition of the VMDK file
#VMDSIZE is the size of the Virtual Disk File being created ex (500mb) or (10g)
####################################
VMNAME="vm_name"
VMOS="winNetStandard"
VMMEMSIZE="256"
DESTVMFS="vmhba0:6:0:1 #Must use the vmhba path
VMDSIZE="10g"
#####END MODIFICATION#####
LOG="/opt/altiris/deployment/adlagent/bin/logevent"
$LOG -l:1 -ss:"Creating VMX Configuration File"
mkdir /home/vmware/$VMNAME
exec 6>&1
exec 1>/home/vmware/$VMNAME/$VMNAME.vmx
# write the configuration file
echo #!/usr/bin/vmware
echo config.version = '"'6'"'
echo virtualHW.version = '"'3'"'
echo memsize = '"'$VMMEMSIZE'"'
echo floppy0.present = '"'TRUE'"'
echo usb.present = '"'FALSE'"'
echo displayName = '"'$VMNAME'"'
echo guestOS = '"'$VMOS'"'
echo suspend.Directory = '"'/vmfs/vmhba0:0:0:5/'"'
echo checkpoint.cptConfigName = '"'$VMNAME'"'
echo priority.grabbed = '"'normal'"'
echo priority.ungrabbed = '"'normal'"'
echo ide1:0.present = '"'TRUE'"'
echo ide1:0.fileName = '"'auto detect'"'
echo ide1:0.deviceType = '"'cdrom-raw'"'
echo ide1:0.startConnected = '"'FALSE'"'
echo floppy0.startConnected = '"'FALSE'"'
echo floppy0.fileName = '"'/dev/fd0'"'
echo Ethernet0.present = '"'TRUE'"'
echo Ethernet0.connectionType = '"'monitor_dev'"'
echo Ethernet0.networkName = '"'Network0'"'
echo draw = '"'gdi'"'
echo
```

```
echo scsi0.present = '"'TRUE'"'
echo scsi0:1.present = '"'TRUE'"'
echo scsi0:1.name = '"'vmhba0:0:0:5:$VMNAME.vmdk'"'
echo scsi0:1.writeThrough = '"'TRUE'"'
echo scsi0.virtualDev = '"'vmxlsilogic'"'
echo
# close file
exec 1>&-
# make stdout a copy of FD 6 (reset stdout), and close FD6
exec 1>&6
exec 6>&-
$LOG -l:1 -ss:"VMX Configuration File Created Successfully"
#Change the file permissions
chmod 755 /home/vmware/$VMNAME/$VMNAME.vmx
#Create the Virtual Disk
$LOG -l:1 -ss:"Creating Virtual Disk"
vmkfstools -c $VMDSIZE vmhba0:0:0:5:$VMNAME.vmdk
$LOG -l:1 -ss:"Virtual Disk Created Successfully"
#Register the new VM
$LOG -l:1 -ss:"Registering VMX Configuration"
#Registering .vmx Configuration"
vmware-cmd -s register /home/vmware/$VMNAME/$VMNAME.vmx
$LOG -l:1 -ss:"VMX Initialization Completed Successfully"
#Starting the Virtual Machine
$LOG -l:1 -ss:"Starting the Virtual Machine"
vmware-cmd /home/vmware/$VMNAME/$VMNAME.vmx start
$LOG -l:1 -ss:"Virtual Machine Started"
$LOG -l:1 -ss:"Passing control to Altiris for PXE boot and install of VM"
```

Let's make one more change to the script so that when the virtual machine first boots up with a brand-new disk, it will boot from the virtual CD-ROM that has an ISO file mounted to it (see Code Listing 9.11).

Code Listing 9.11 Creating a New Virtual Machine That Boots to an ISO

```
#!/bin/bash
#Scripting VMware Power Tools: Automating Virtual Infrastructure Administration
#Creates a new Virtual Machine booting to an ISO
#Stephen Beaver
#####USER MODIFICATION#################
#VMNAME is the name of the new virtual machine
#VMOS specifies which Operating System the virtual machine will have
```

```
#GLDIMAGE is the path to the "Golden Image" VMDK file
#DESTVMFS is the path to the VMFS partition of the VMDK file
#VMDSIZE is the size of the Virtual Disk File being created ex (500mb) or (10g)
#ISOIMAGE is the path and file name of the ISO file you are using
#####################################
VMOS="winNetStandard"
VMMEMSIZE="256"
GLDIMAGE="/vmfs/FHVMFS1/Windows_2003_Standard.vmdk"
DESTVMFS="vmhba0:0:0:10"
VMDSIZE="10g"
ISOIMAGE"/vmfs/ESX_SAN/Windows2000.iso"
#####END MODIFICATION#####
LOG="/var/log/$1.log"
echo "Start of Logging" > $LOG
echo "Importing Golden Image Disk File VMDK" >> $LOG
vmkfstools -i $GLDIMAGE $DESTVMFS:$1.vmdk
echo "Creating VMX Configuration File" >> $LOG
mkdir /home/vmware/$1
exec 6>&1
exec 1>/home/vmware/$1/$1.vmx
# write the configuration file
echo #!/usr/bin/vmware
echo config.version = '"'6'"'
echo virtualHW.version = '"'3'"'
echo memsize = '"'$VMMEMSIZE'"'
echo floppy0.present = '"'TRUE'"'
echo usb.present = '"'FALSE'"'
echo displayName = '"'$1'"'
echo guestOS = '"'$VMOS'"'
echo suspend.Directory = '"'/vmfs/vmhba0:0:0:10/'"'
echo checkpoint.cptConfigName = '"'$1'"'
echo priority.grabbed = '"'normal'"'
echo priority.ungrabbed = '"'normal'"'
echo ide1:0.present = '"'TRUE'"'
echo ide0:0.present = '"'TRUE'"'
echo ide0:0.fileName = '"'$ISOIMAGE'"'
echo ide0:0.deviceType = '"'cdrom-image'"'
echo floppy0.startConnected = '"'FALSE'"'
echo floppy0.fileName = '"'/dev/fd0'"'
echo Ethernet0.present = '"'TRUE'"'
```

```
echo Ethernet0.connectionType = '"'monitor_dev'"'
echo Ethernet0.networkName = '"'Network0'"'
echo draw = '"'gdi'"'
echo
echo scsi0.present = '"'TRUE'"'
echo scsi0:1.present = '"'TRUE'"'
echo scsi0:1.name = '"'$DESTVMFS:$1.vmdk'"'
echo scsi0:1.writeThrough = '"'TRUE'"'
echo scsi0.virtualDev = '"'vmxlsilogic'"'
echo
# close file
exec 1>&-
# make stdout a copy of FD 6 (reset stdout), and close FD6
exec 1>&6
exec 6>&-
#Create the Virtual Disk
echo "Creating Virtual Disk" >> $LOG
vmkfstools -c $VMDSIZE vmhba0:0:0:5:$VMNAME.vmdk
echo "Virtual Disk Created Successfully" >> $LOG
echo "VMX Configuration File Created Successfully" >> $LOG
#Change the file permissions
chmod 755 /home/vmware/$1/$1.vmx
#Register the new VM
echo "Registering .vmx Configuration" >> $LOG
vmware-cmd -s register /home/vmware/$1/$1.vmx
echo "VMX Initialization Completed Successfully" >> $LOG
#Starting the Virtual Machine
echo "Starting the Virtual Machine" >> $LOG
vmware-cmd /home/vmware/$VMNAME/$VMNAME.vmx start
echo "Virtual Machine Started" >> $LOG
```

Summary

Let's review what we've covered. First, we took a solid look at the virtual disk files (*.vmdk). We opened up the disk descriptor file, reviewed its contents, and converted an IDE virtual disk file to a SCSI virtual disk file. We then took an in-depth look at the settings inside the virtual machine configuration files (*.vmx) and finished the IDE-to-SCSI conversion.

I presented a few scripts that covered backing up the configuration files of the virtual machines, and how to build virtual machines. I also discussed a few options for making changes to all (or one) virtual machines at the same time. You can use bits and parts of these different scripts to open the door to various types of automation. Using the native "sed" program, for example, you have the ability to script the edits to any of the files you need. This gives you a wide range of options that can be scripted and automated. The vmware-cmd tool also opens a lot of doors thanks to the different choices available. Run vmware-cmd from the service console to view all the options and syntax.