

Chapter 4

Building a VM

Topics in this chapter:

- **Creation of Virtual Machines Utilizing Command-Line Tools**
- **Scripting Creation of Virtual Machines in ESX Shell**
- **Scripting Creation of Virtual Machines in Perl Scripts**
- **Cloning Virtual Machines Utilizing ESX Shell Scripts**
- **Cloning Virtual Machines Utilizing VmPerl Scripts**

Introduction

VMware provides many useful command-line tools for the creation and cloning of virtual machines. In this chapter, you will gain a working understanding of these tools and how to leverage them to automate virtual machine creation. At the end of this chapter, you will be able to script the creation and cloning of virtual machines to automate your virtual machine setup.

Creation of Virtual Machines Utilizing Command-Line Tools

VMware ESX Server has tools available for command-line creation and cloning of virtual machines. These tools are available via the service console and require that you access the service console with root-level privileges.



TIP

Remote access to ESX Server by default is disabled for the root account. Create an additional account on your ESX server. Log in with this new account and use the **su – root** command. This command will allow you to assume root level privileges.

Three main steps are involved to create a virtual machine utilizing the command-line tools.

- Creation of a virtual machine configuration file
- Creation of a virtual machine disk file
- Registering the virtual machine with ESX Server

We accomplish the preceding tasks by utilizing the following tools:

- text editor such as VI
- *vmkfstools*
- *vmware-cmd*

Creation of a Virtual Machine Configuration File

Virtual machine configurations are stored as files with a .vmx extension. The VMX file is just a text file with specific fields that define the virtual machine's configuration. A very short vmx file only needs 14 lines to support a virtual machine that encompasses one CPU, one hard drive, and one network adapter. You could create a VMX file with just three lines but it would be of minimal value. Code Listings 4.1 and 4.2 show sample VMX configurations.

Code Listing 4.1 ESX 2.x VMX Code

```
guestOS = "winnetenterprise"
config.version = "6"
virtualHW.version = "3"

scsi0.present = "true"
scsi0.sharedBus = "none"
scsi0.virtualDev = "lsilogic"

memsize = "512"
scsi0:0.present = "true"
scsi0:0.fileName = "ESX Created VM.vmdk"
scsi0:0.deviceType = "scsi-hardDisk"

ethernet0.present = "true"
ethernet0.allowGuestConnectionControl = "false"
ethernet0.networkName = "VM Network"
ethernet0.addressType = "vpx"
```

Code Listing 4.2 ESX 3.x VMX Code

```
guestOS = "winnetenterprise"
config.version = "8"
virtualHW.version = "4"

scsi0.present = "true"
scsi0.sharedBus = "none"
scsi0.virtualDev = "lsilogic"

memsize = "512"
```

142 Chapter 4 • Building a VM

```
scsi0:0.present = "true"
scsi0:0.fileName = "ESX Created VM.vmdk"
scsi0:0.deviceType = "scsi-hardDisk"
ethernet0.present = "true"
ethernet0.allowGuestConnectionControl = "false"
ethernet0.networkName = "VM Network"
ethernet0.addressType = "vpx"
```

As you can tell from Code Listings 4.1 and 4.2, the only difference is in the values of the `config.version` and `virtualHW.version` entries. These values relate to the version of ESX Server you are running. To check the values for these fields, open up an existing virtual machine's configuration file in a text editor.

NOTE

It doesn't matter whether you use upper- or lowercase, but always make sure to use " " for the values in an (VMX) file.

Once you start a VM using a VMX configuration file like the ones shown in Code Listings 4.1 and 4.2, VMware will generate additional entries in the VMX. These entries identify the virtual machine and set default values for the virtual machine. Examples of these types of entries are shown in Code Listing 4.3

Code Listing 4.3 VMware Autogenerated VMX Entry Examples

```
uuid.bios = "56 4d ee 3c 52 06 a3 de-be 4a 73 9c cc 79 25 2b "
ethernet0.generatedAddress = "00:50:56:a7:42:e2"
powerType.powerOff = "default"
powerType.powerOn = "default"
powerType.suspend = "default"
powerType.reset = "default"
```

NOTE

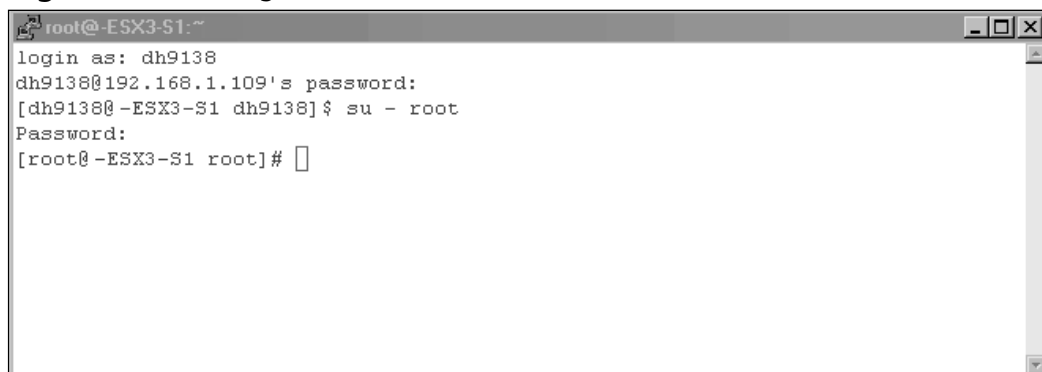
Configuration files for virtual machines created with VMware ESX Server 2.0 and later use the .vmx extension. Earlier versions of ESX Server used the .cfg extension.

Creating Your Virtual Machine Configuration File

You now have a basic understanding of how a virtual machine configuration file is constructed and are ready to build your own. The steps that follow detail how to create a new virtual machine configuration file.

- Log in locally or connect to your ESX server remotely.
- Log in with an ID that has root privileges (see the Tip in the previous section), as shown in Figure 4.1.

Figure 4.1 Gaining Root Level Access on ESX Server

A terminal window titled 'root@-ESX3-S1:~' showing a login process. The user 'dh9138' logs in from IP '192.168.1.109'. The prompt is '[dh9138@-ESX3-S1 dh9138]\$. The user enters 'su - root' and provides a password. The prompt changes to '[root@-ESX3-S1 root]#'.

```
root@-ESX3-S1:~
login as: dh9138
dh9138@192.168.1.109's password:
[dh9138@-ESX3-S1 dh9138]$ su - root
Password:
[root@-ESX3-S1 root]#
```

- Change to the location of where you want to put your new virtual machine. Virtual machine configuration files (VMX) have to be stored in the same location as the other virtual machine files (VSWP, VMDK, and so on). See Figure 4.2.

Figure 4.4 Creating a New Virtual Machine Configuration File in VI

- Press **I** to turn on inserting (you will see the word insert at the bottom of the screen).
- Type in the following example virtual machine configuration file (see Code Listing 4.4).

Code Listing 4.4 Example Virtual Machine Configuration File

```
config.version = "6"  
virtualHW.version = "3"  
memsize = "256"  
floppy0.present = "false"  
displayName = "newVM"  
guestOS = "winNetStandard"  
ide0:0.present = "TRUE"  
ide0:0.deviceType = "cdrom-raw"  
ide0:0.startConnected = "false"  
floppy0.startConnected = "FALSE"  
floppy0.fileName = "/dev/fd0"  
Ethernet0.present = "TRUE"  
Ethernet0.connectionType = "monitor_dev"  
Ethernet0.networkName = "VM Network"  
Ethernet0.addressType = "vpx"  
scsi0.present = "true"  
scsi0.sharedBus = "none"
```

146 Chapter 4 • Building a VM

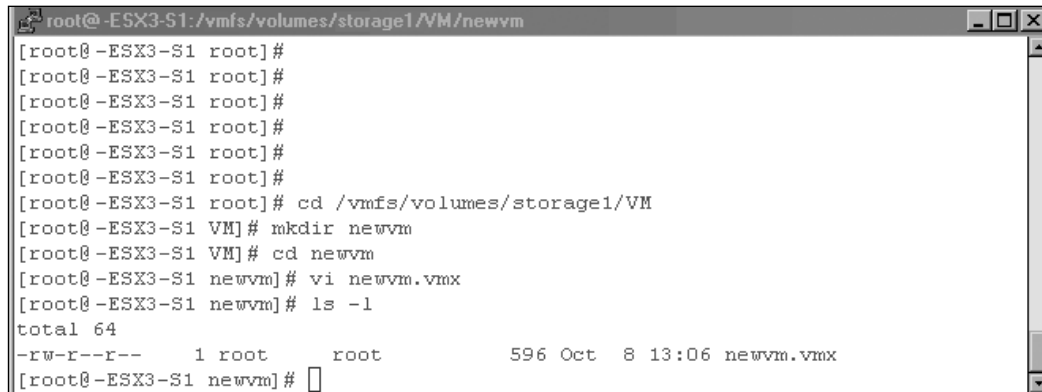
```
scsi0.virtualDev = "lsilogic"
scsi0:0.present = "true"
scsi0:0.fileName = "newvm.vmdk"
scsi0:0.deviceType = "scsi-hardDisk"
```

- Press the **Esc** key to exit the insert mode, then press and hold **Shift** and press **ZZ** to save and exit (see Figure 4.5).

Figure 4.5 Saving the VMX File in VI

```
root@-ESX3-S1:/vms/volumes/storage1/VM/newvm
config.version = "8"
virtualHW.version = "4"
memsize = "256"
floppy0.present = "false"
displayName = "newVM"
guestOS = "winNetStandard"
ide0:0.present = "TRUE"
ide0:0.deviceType = "cdrom-raw"
ide0:0.startConnected = "false"
floppy0.startConnected = "FALSE"
floppy0.fileName = "/dev/fd0"
Ethernet0.present = "TRUE"
Ethernet0.connectionType = "monitor_dev"
Ethernet0.networkName = "VM Network"
Ethernet0.addressType = "vpx"
scsi0.present = "true"
scsi0.sharedBus = "none"
scsi0.virtualDev = "lsilogic"
scsi0:0.present = "true"
scsi0:0.fileName = "newvm.vmdk"
scsi0:0.deviceType = "scsi-hardDisk"
-- INSERT --
```

- Type **ls -l** to get a directory listing. You should now see your new virtual machine configuration file (see Figure 4.6).

Figure 4.6 Completed Creation of VMX FileA terminal window screenshot showing a series of commands and their outputs. The user is in the directory /vmfs/volumes/storage1/VM/newvm. The commands executed are: cd /vmfs/volumes/storage1/VM, mkdir newvm, cd newvm, vi newvm.vmx, and ls -l. The output of ls -l shows a file named newvm.vmx with permissions -rw-r--r--, size 596, and creation date Oct 8 13:06.

```
root@-ESX3-S1:/vmfs/volumes/storage1/VM/newvm
[root@-ESX3-S1 root]#
[root@-ESX3-S1 root]#
[root@-ESX3-S1 root]#
[root@-ESX3-S1 root]#
[root@-ESX3-S1 root]#
[root@-ESX3-S1 root]#
[root@-ESX3-S1 root]#
[root@-ESX3-S1 root]# cd /vmfs/volumes/storage1/VM
[root@-ESX3-S1 VM]# mkdir newvm
[root@-ESX3-S1 VM]# cd newvm
[root@-ESX3-S1 newvm]# vi newvm.vmx
[root@-ESX3-S1 newvm]# ls -l
total 64
-rw-r--r--  1 root   root           596 Oct  8 13:06 newvm.vmx
[root@-ESX3-S1 newvm]#
```

You are now ready to go on to the next section to create the virtual disk `newvm.vmdk` that you will be referencing in your configuration file.

NOTE

Do not log out or close your ESX session just yet. You will continue from this point in the next section.

Creation of a Virtual Machine Disk File

VMware has a command-line utility, called *vmkfstools*, which can be used for the creation of VMFS file systems and virtual machine disk files. In this chapter, we will only focus on the options that pertain to virtual disks. For a full listing of command options, type **vmkfstools** in a console session or **man vmkfstools**. Code Listing 4.5 lists the *vmkfstools* options that pertain to virtual disks.

Code Listing 4.5 *vmkfstools* Command Options for Virtual Disks

```
vmkfstools
OPTIONS FOR VIRTUAL DISKS:
vmkfstools -c --createvirtualdisk #[gGmMkK]
              -d --diskformat [zeroedthick]
```

148 Chapter 4 • Building a VM

```

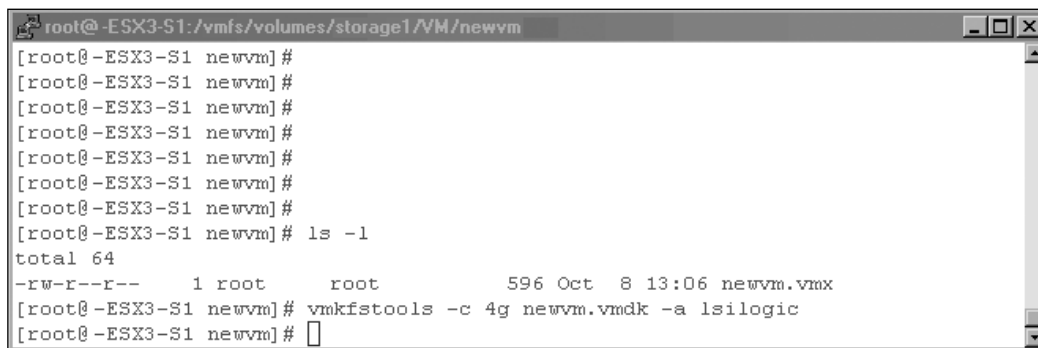
eagerzeroedthick|
thick|
thin]
-a --adapterType [buslogic|lsilogic]
-w --writezeros
-j --inflatedisk
-U --deletevirtualdisk
-E --renamevirtualdisk srcDisk
-i --clonevirtualdisk srcDisk
-d --diskformat [rdm:<device>|rdmp:<device>|
raw:<device>|thin|2gbsparse]
-X --extendvirtualdisk #[GmMkK]
-M --migratevirtualdisk
-r --createrdm /vmfs/devices/disks/...
-q --queryrdm
-z --createrdmpassthru /vmfs/devices/disks/...
-Q --createrawdevice /vmfs/devices/generic/...
-v --verbose #
-g --geometry

vmfsPath

```

In our example, we will create a 4GB virtual disk called `newvm.vmdk` and assign it a SCSI LSI Logic adapter. In the console, type the following: **vmkfstools -c 4g newvm.vmdk -a lsilogic**. Then, press **Enter** (see Figure 4.7).

Figure 4.7 Creating the Virtual Disk



```

root@-ESX3-S1:/vmfs/volumes/storage1/VM/newvm
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]# ls -l
total 64
-rw-r--r--  1 root   root      596 Oct  8 13:06 newvm.vmx
[root@-ESX3-S1 newvm]# vmkfstools -c 4g newvm.vmdk -a lsilogic
[root@-ESX3-S1 newvm]#

```

We have now created a virtual disk file `newvm.vmdk` in the same location as our virtual machine configuration file. The last step is to register this new virtual machine with ESX Server.

Registering Virtual Machines with ESX Server

VMware includes the `vmware-cmd` command tool for performing various operations on virtual machines and the server. In this chapter, we will focus on the virtual machine registration option of this tool `-s register`. For more information on all available tool options, type **vmware-cmd** at the console command prompt.

Type the following all on one line in the console window to register the new virtual machine with the ESX server:

```
vmware-cmd -s register "Your Path"/newvm/newvm.vmx
```

“Your Path” should be in a similar format to `/vmfs/volumes/storage1/` (see Figure 4.8).

Figure 4.8 Registering a Virtual Machine with ESX



```
root@-ESX3-S1:/vmfs/volumes/storage1/VM/newvm
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]#
[root@-ESX3-S1 newvm]# vmware-cmd -s register /vmfs/volumes/storage1/VM/newvm
/newvm.vmx
register (/vmfs/volumes/storage1/VM/newvm/newvm.vmx) = 1
[root@-ESX3-S1 newvm]#
```

A returned value of “1” after running this command indicates a successful registration of the virtual machine. Open up the GUI of your ESX server to verify that the new VM is listed. At this point, you are ready to power on the virtual machine and install your operating system. You have successfully created a new virtual machine utilizing the VMware command-line tools.

WARNING

If when turning on the virtual machine for the first time you receive the error message “Cannot open disk <diskname.vmdk>:The system cannot find the file specified. Bad value for scsi0:0.virtualDev,” this means your virtual machine configuration file has the wrong values for config.version and virtualHW.version. Update the values for these two fields with the appropriate ones for your version of ESX.

Scripting Creation of Virtual Machines in ESX Shell

Scripting the creation of virtual machines is simpler than one might think. We will leverage your new gained experience from previous sections on utilizing the VMware tools to automate the VM creation process. In the previous section, you manually created a virtual machine configuration file and virtual disk. You then registered the virtual machine with ESX Server. We’re going to now essentially take all those commands and steps and automate them in a script that you can run repeatedly and customize to build various types of virtual machines.

The VMware ESX shell is simply the service console operating system. This operating system is a custom version of Linux that VMware created. In Linux, you can create a simple text file to automate commands and then execute it. If you are familiar with DOS batch files, then this will be easy for you. Code Listing 4.6 shows an example of scripted VM creation.

Code Listing 4.6 Scripted VM Creation

```
##### VM Creation Script #####
#Script Version 1.1
#Author David E. Hart
#Date 10-05-06
#
#-----+
# Purpose|
```

```

#-----+-----
# This script will create a VM with the following attributes;
# Virtual Machine Name = ScriptedVM
# Location of Virtual Machine = /VMFS/volumes/storage1/ScriptedVM
# Virtual Machine Type = "Microsoft Windows 2003 Standard"
# Virtual Machine Memory Allocation = 256 meg
#
#-----+-----
#Custom Variable Section for Modification|
#-----+-----
#NVM is name of virtual machine(NVM). No Spaces allowed in name
#NVMDIR is the directory which holds all the VM files
#NVMOS specifies VM Operating System
#NVMSIZE is the size of the virtual disk to be created
#-----+-----
#####

### Default Variable settings - change this to your preferences
NVM="ScriptedVM"      # Name of Virtual Machine
NVMDIR="ScriptedVM"  # Specify only the folder name to be created; NOT the
complete path
NVMOS="winnetstandard" # Type of OS for Virtual Machine
NVMSIZE="4g"          # Size of Virtual Machine Disk
VMMEMSIZE="256"      # Default Memory Size

### End Variable Declaration

mkdir /vmfs/volumes/storage1/$NVMDIR # Creates directory
exec 6>&1                               # Sets up write to file
exec 1>/vmfs/volumes/storage1/$NVMDIR/$NVM.vmx # Open file
# write the configuration
echo config.version = "'6'"           # For ESX 3.x the value is 8
echo virtualHW.version = "'3'"        # For ESX 3.x the value is 4
echo memsize = "'$VMMEMSIZE'"

```

152 Chapter 4 • Building a VM

```
echo floppy0.present = ''TRUE'' # setup VM with floppy
echo displayName = ''$NVM'' # name of virtual machine
echo guestOS = ''$NVMOS''
echo
echo ide0:0.present = ''TRUE''
echo ide0:0.deviceType = ''cdrom-raw''
echo ide0:0.startConnected = ''false'' # CDROM enabled
echo floppy0.startConnected = ''FALSE''
echo floppy0.fileName = ''/dev/fd0''
echo Ethernet0.present = ''TRUE''
echo Ethernet0.networkName = ''VM Network'' # Default network
echo Ethernet0.addressType = ''vpx''
echo
echo scsi0.present = ''true''
echo scsi0.sharedBus = ''none''
echo scsi0.virtualDev = ''lsilogic''
echo scsi0:0.present = ''true'' # Virtual Disk Settings
echo scsi0:0.fileName = ''$NVM.vmdk''
echo scsi0:0.deviceType = ''scsi-hardDisk''

echo
# close file
exec 1>&-

# make stdout a copy of FD 6 (reset stdout), and close FD6
exec 1>&6
exec 6>&-

# Change permissions on the file so it can be executed by anyone
chmod 755 /vmfs/volumes/storage1/$NVMDIR/$NVM.vmx

#Creates 4gb Virtual disk
cd /vmfs/volumes/storage1/$NVMDIR #change to the VM dir
```

```
vmkfstools -c $NVMSIZE $NVM.vmdk -a lsilogic

#Register VM
vmware-cmd -s register /vmfs/volumes/storage1/$NVMDIR/$NVM.vmx
```

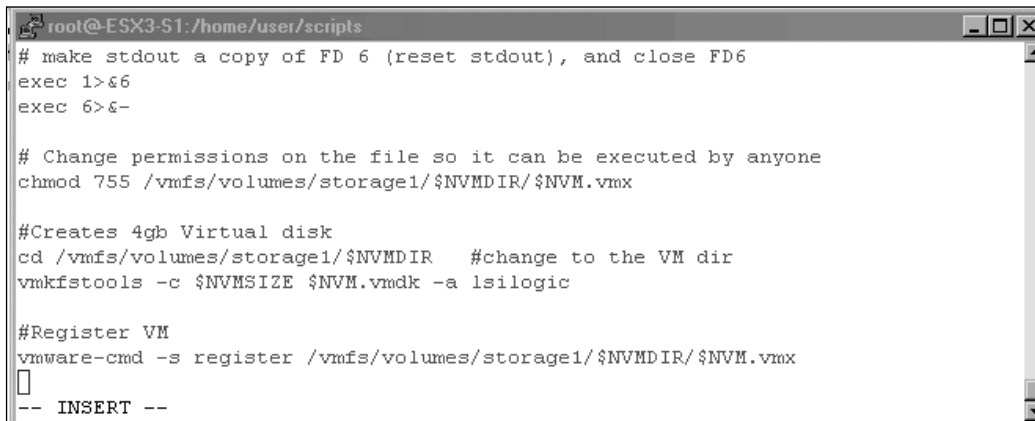
NOTE

The standard format for values for the VMX file are to encase them in double quotes, such as `memsize = "256"`. When scripting the creation of the VMX, you need to use single quote, double quote, single quote. So the previous example would be `memsize = "'256'"`. You must do this for VMX values.

The script in Code Listing 4.6 will create a virtual machine that has the following characteristics:

- A VM called `ScriptedVM` in a directory named `ScriptedVM` on `storage1`
- A VM that will be assigned 256MB of memory
- A VM that will have a 4GB SCSI hard drive (lsilogic controller)
- A VM configured for a Windows 2003 standard operating system
- A floppy drive assigned, not connected at startup
- A CD-ROM attached to the ESX server's CD-ROM drive, not connected at startup
- An Ethernet adapter connected to the VM Network, enabled at startup

The `exec` commands in the script are system-level commands in Linux to set up the writing to, and saving of, the script file. It redirects the console screen's output to the script file. The use of the `echo` commands in the script sends the commands to the screen which are redirected to the file for writing. The file is then closed and the virtual configuration file, VMX, is saved. The permissions are changed on the configuration file so any user on ESX can access the virtual machine. Then the script creates the virtual disk and registers the VM with the ESX server.

Figure 4.11 Using VI to Create Shell Script


```

root@-ESX3-S1:/home/user/scripts
# make stdout a copy of FD 6 (reset stdout), and close FD6
exec 1>&6
exec 6>&-

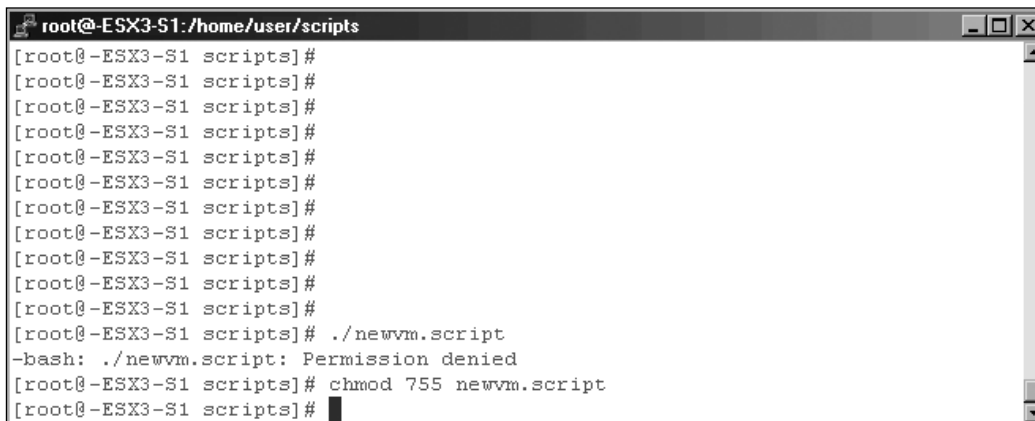
# Change permissions on the file so it can be executed by anyone
chmod 755 /vmfs/volumes/storage1/$NVMDIR/$NVM.vmx

#Creates 4gb Virtual disk
cd /vmfs/volumes/storage1/$NVMDIR #change to the VM dir
vmkfstools -c $NVMSIZE $NVM.vmdk -a lsilogic

#Register VM
vmware-cmd -s register /vmfs/volumes/storage1/$NVMDIR/$NVM.vmx
[
-- INSERT --

```

- Press **Esc** and then press and hold **Shift** while pressing **ZZ** to exit and save.
- You should now have a file called `newvm.script` listed. Before you run the script, you must set permissions on it. To do this, type **chmod 755 newvm.script** (see Figure 4.12).

Figure 4.12 Setting Permissions on Script File


```

root@-ESX3-S1:/home/user/scripts
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]#
[root@-ESX3-S1 scripts]# ./newvm.script
-bash: ./newvm.script: Permission denied
[root@-ESX3-S1 scripts]# chmod 755 newvm.script
[root@-ESX3-S1 scripts]# █

```

- Run the script by typing **./newvm.script** (see Figure 4.13).

Figure 4.13 The Execution of Shell Script

```

root@-ESX3-S1:/home/user/scripts
[root@-ESX3-S1 storage1]#
[root@-ESX3-S1 storage1]#
[root@-ESX3-S1 storage1]# cd /
[root@-ESX3-S1 /]# cd home
[root@-ESX3-S1 home]# ls
dh9138  lost+found  user
[root@-ESX3-S1 home]# cd user
[root@-ESX3-S1 user]# ls
scripts
[root@-ESX3-S1 user]# cd scripts
[root@-ESX3-S1 scripts]# ls
newvm.script
[root@-ESX3-S1 scripts]# ./newvm.script
register (/vmfs/volumes/storage1/ScriptedVM/ScriptedVM.vmx) = 1
[root@-ESX3-S1 scripts]#

```

The virtual machine has now been created and registered with ESX. The next steps are for you to power it on and install the guest operating system. Creating scripts in ESX shell will save you time and effort in creating new virtual machines in your environment.

TIP

VMware ESX shell is just a customized version of Linux. For more information on scripting in shell, reference Linux shell information and examples.

Swiss Army Knife...**Creating Templates with the Scripted VM Creation Script**

You can create multiple copies of the Scripted VM Creation script all with unique configurations. Save each of these customized scripts with a descriptive name such as **2003std512m4g.script** or **2003ent1g4g.script**. You could use the ESX shell command **cp Source.script Target.script** to copy the first script and then use **VI** to customize the second one. Change

Continued

each script to store its VMs in a staging area and each VM with a unique name. Now when you need to build those types of VMs you have template scripts to do it with. You could even chain together running of these scripts so you can create complete virtual system setups.

If you are not comfortable with utilizing VI as a text editor, you could also use a text editor such as Notepad on your PC to create your script. Once you have completed your script, highlight the whole script and select **copy**. Connect to your ESX server with a tool like **Putty** and run the **VI** command. Select **I** for input, and then paste in your script. All the script examples in this book were created on ESX Server using that method. Alternatively, you could also create the script file locally and then use a tool like WinSCP to copy the file to your ESX server.

Scripting Creation of Virtual Machines in Perl Scripts

VMware ESX Server supports additional scripting languages such as VmPerl. VmPerl is VMware's version of the Perl programming language. VMware has designed VmPerl to provide task automation and simple, single-purpose user interfaces. VmPerl's main purpose is to interact with the virtual machines on ESX Server. You can query status, start and stop virtual machines, as well as manage snapshots. With some creative scripting we can have VmPerl create our virtual machines for us as well. Scripting in VmPerl is not for beginners. If you've never scripted in Perl before then review the sample VmPerl Script and note the code comments in the script. VmPerl is a customized version of Perl, so research the Perl language in general for more information on how to program in Perl. The example script in Code Listing 4.7 was written so it could be easily modified to suit your particular needs. It is a basic VmPerl script with a menu-driven interface. Leveraging the knowledge you've gained in the previous sections will help you understand the script interactions. Novices in scripting should find the next script example very easy to understand and follow. Experienced scripters may find the script rudimentary and know of alternate ways to accomplish similar tasks. Whatever your scripting experience, I hope you find the example scripts in this chapter thought provoking and insightful.

NOTE

VMware in its latest release of ESX Server 3.0 is moving toward more mainstream scripting languages. Scripting APIs such as VmCom and VmPerl are being deprecated. What this means is that VMware prefers that you use a new set of APIs for programming languages such as Java, Visual .NET, and so on. ESX 3.0 will continue to support VmPerl and ESX shell scripting, and all the sample scripts in the chapter have been tested on ESX 3.0.

VmPerl allows for flexibility on how you go about creating your virtual machines. In our example script (see Code Listing 4.7), I used VmPerl's user input and file manipulation commands to accomplish the three primary tasks when creating virtual machines.

- Creating the virtual machine configuration file (VMX)
- Creating the virtual machine disk file (VMDK)
- Registering the virtual machine with ESX Server

NOTE

The script shown in Code Listing 4.7 is meant to be used for educational purposes. Further development in the area of "error checking" and "handling" should be done prior to utilizing it in a production environment. It is meant only to show what can be done with VmPerl.

Code Listing 4.7 Scripted VM Creation with Perl

```
#!/usr/bin/perl -w

use VMware::VmPerl;
use VMware::VmPerl::Server;
use VMware::VmPerl::ConnectParams;
#use strict;
```

```
##### VM Menu Driven Creation Script #####
#Script Version 1.8
#Author David E. Hart
#Date 10-05-06
#
#-----+
#Purpose |
#-----+
# This script presents a menu for automatically building
# virtual machine config files (VMX) and Dis files (VMDK)
# This script demonstrates how to automate the setup
# of virtual environments
#-----+
#Custom Variables Section |
#-----+
#vmname = virtual machine name, will be used for disk as well
#vmmem = amount of memory assigned to VM
#vmos = OS that VM is configured for
#vmdisk = size of VM disk
#####

main:      # main menu

system("clear");
print "                MAIN MENU \n";
print "----- Virtual Machine Creation ----- \n";
print "\n";
print "\n";
print "\n";
print "                1) Create a Custom VM \n";
print "\n";
print "                2) Create VM's from Defined Templates \n";
print "\n";
```

160 Chapter 4 • Building a VM

```
print "          3) View ESX's registered VM's \n";
print "\n";
print "          4) Exit \n";
print "\n";
print "    Your Selection - ";
$menuopt = <>; chomp $menuopt;      # Get user selection
if ($menuopt == 1) { # Get input for custom VM
    system("clear");
    print "What do you want to name your VM? ";
    $vmname = <>; chomp $vmname;    # use chomp to remove carriage return
    print "How much memory do you want to assign? ";
    $vmmem = <>;chomp $vmmem;
    print "Do you want to run Windows 2003STD as the OS? (y/n) ";
    $vmos = <>;chomp $vmos;
    if ($vmos eq "y") {
        $vmos = "winNetStandard";
    } # Only 2 options for this example
    else {
        print "Do you want to run Windows 2003Ent as the OS? (y/n) ";
        $vmos2 = <>;chomp $vmos2;
        if ($vmos2 eq "y") {
            $vmos = "winnetenterprise";
        }
    }
    print "What size hard disk do you want to set up (gb)? ";
    $vmdisk = <>;chomp $vmdisk;
    print "\n";
    $x = writevmx(); # Subrouting for creating VMX file
    if ($x == 1) {
        print "VMX File written successfully \n";
    }
    $w = setper(); # Subroutine to set permissions so anyone can
    use VM
    if ($w == 1) {
```

```
        print "Permissions set successfully \n";
    }
    $y = createdisk(); # subrouting to create VMDK disk file
    if ($y == 1) {
        print "Virtual disk created successfully \n";
    }
    $z = registervm(); # subroutine to register VM with ESX
    if ($z == 1) {
        print "VM registered successfully \n";
    }
    print "Press the ENTER key to continue ...";
    $pause = <STDIN>;
    goto main
    }

if ($menuopt == 2) { # option to displays the templates
menu1:
    system("clear");
    print "                Defined Templates \n";
    print "                ----- \n";
    print "\n";
    print "\n";
    print "            1) Windows 2003std VM with 256m, 4gb drive \n";
    print "\n";
    print "            2) Windows 2003ent VM with 1gig, 8gb drive \n";
    print "\n";
    print "\n";
    print "\n";
    print "\n";
    print "    Your Selection - ";
    $menuopt = <>; chomp $menuopt;
    if ($menuopt == 1) {
        $vmname = "2003std25m4gb";
        $vmmem = "256"; # change and add on similar sections
    }
}
```

162 Chapter 4 • Building a VM

```
$vmdisk = "4"; # to create templates for your environment
$vmos = "winnetstandard";
$x = writevmx();
  if ($x == 1) {
    print "VMX File written successfully \na";
  }
$w = setper();
  if ($w == 1) {
    print "Permissions set successfully \na";
  }
$y = createdisk(); # Call subroutines to create VMs
  if ($y == 1) {
    print "Virtual disk created successfully \na";
  }
$z = registervm();
  if ($z == 1) {
    print "VM registered successfully \na";
  }
print "Press the ENTER key to continue ...";
$pause = <STDIN>;
goto main
}
if ($menuopt == 2) {
  $vmname = "2003Ent1gb8gb";
  $vmmem = "1024";
  $vmdisk = "8";
  $vmos = "winnetenterprise";
  $x = writevmx();
  if ($x == 1) {
    print "VMX file written successfully \na";
  }
  $w = setper();
  if ($w == 1) {
    print "Permissions set successfully \na";
```

```
    }
    $y = createdisk();
    if ($y == 1) {
        print "Virtual disk created successfully \na";
    }
    $z = registervm();
    if ($z == 1) {
        print "VM registered successfully \na";
    }
    print "Press the ENTER key to continue ...";
    $pause = <STDIN>;
    goto main
}
else {
    goto menu1;
}

}

if ($menuopt == 3) {    # Use a function of VmPerl to display registered VMs
    system("clear");
    my ($server_name, $user, $passwd) = @ARGV; # Assume running in ESX
server
    my $port = 902; # with appropriate
rights

VMware::VmPerl::ConnectParams::new($server_name,$port,$user,$passwd);
VMware::VmPerl::ConnectParams::new(undef,$port,$user,$passwd);
my $connect_params = VMware::VmPerl::ConnectParams::new();

# Establish a persistent connection with server
my $server = VMware::VmPerl::Server::new();
if (!$server->connect($connect_params)) {
    my ($error_number, $error_string) = $server->get_last_error();
```

164 Chapter 4 • Building a VM

```

        die "Could not connect to server: Error $error_number:
$error_string\n";
    }

    print "\nThe following virtual machines are registered:\n";

    # Obtain a list containing every config file path registered with the
server.
    my @list = $server->registered_vm_names();
    if (!defined($list[0])) {
        my ($error_number, $error_string) = $server->get_last_error();
        die "Could not get list of VMs from server: Error $error_number:
.".
            "$error_string\n";
    }

    print "$_\n" foreach (@list);

    # Destroys the server object, thus disconnecting from the server.
    undef $server;
    print "Press the ENTER key to continue ...";
    $pause = <STDIN>;
    goto main

}

if ($menuopt == 4) {
    goto endl
}

sub writevmx {          # Subroutine to Create VM's VMX config file

#       $file = '/vmfs/volumes/storage1/perlvm/perlvm.vmx';          #
Name the file
    $file = "/vmfs/volumes/storage1/perlvm/" . $vmname . ".vmx";
    open(INFO, ">$file");    # Open for output

```

```
print INFO 'config.version = "6" ' . "\n";
print INFO 'virtualHW.version = "3" ' . "\n";
print INFO 'memsize = "' . $vmmem . '" ' . "\n";
print INFO 'floppy0.present = "TRUE" ' . "\n";
print INFO 'displayName = "' . $vmname . '" ' . "\n";
print INFO 'guestOS = "' . $vmos . '" ' . "\n";
print INFO 'ide0:0.present = "TRUE" ' . "\n";
print INFO 'ide0:0.deviceType = "cdrom-raw" ' . "\n";
print INFO 'ide0:0.startConnected = "false" ' . "\n";
print INFO 'floppy0.startConnected = "FALSE" ' . "\n";
print INFO 'floppy0.fileName = "/dev/fd0" ' . "\n";
print INFO 'Ethernet0.present = "TRUE" ' . "\n";
print INFO 'Ethernet0.connectionType = "monitor_dev" ' . "\n";
print INFO 'Ethernet0.networkName = "VM Network" ' . "\n";
print INFO 'Ethernet0.addressType = "vpx" ' . "\n";
print INFO 'scsi0.present = "true" ' . "\n";
print INFO 'scsi0.sharedBus = "none" ' . "\n";
print INFO 'scsi0.virtualDev = "lsilogic" ' . "\n";
print INFO 'scsi0:0.present = "true" ' . "\n";
print INFO 'scsi0:0.fileName = "' . $vmname . '.vmdk" ' . "\n";
print INFO 'scsi0:0.deviceType = "scsi-hardDisk" ' . "\n";

close(INFO);          # Close the file
}

sub createdisk {      # Subroutine to create virtual disk
    $scr = "vmkfstools -c " . $vmdisk . ".g " . "
/vmfs/volumes/storage1/perlvm/" . $vmname . ".vmdk -a lsilogic";
    system("$scr");
};

sub registervm {     # Subroutine to register VM with ESX Server
```

166 Chapter 4 • Building a VM

```

        $rg = "vmware-cmd -s register /vmfs/volumes/storage1/perlvm/" .
$vmname . ".vmx";
        system("$rg");
    }

sub setper{          # Subroutine to set permission on VMX file
    $pm = "chmod 755 /vmfs/volumes/storage1/perlvm/" . $vmname .
".vmx";
    system("$pm");
}

endl:

```

Modifying Scripted VM Creation with Perl

The script shown in Code Listing 4.7, and later in Code Listing 4.11, provides static mapping for VM creation. This is sufficient for an example, but not very practical for real-world scenarios. We will modify the script to support end-user input of VM destination pathing. We will accomplish this by adding a new variable *\$vmpath* to our script and adding the appropriate following sections.

- Add new variable *vmpath* to scripts variable notes section

```

#-----+
#Custom Variables Section |
#-----+
#vmname = virtual machine name, used for disk as well
#vmmem = amount of memory assigned to VM
#vmos = OS that VM is configured for
#vmdisk = size of VM disk
#vmpath = path to VM directory, (must already exist)
#####

```

- Add new prompt in Custom VM Creation section, “option 1.”

```
print "What size hard disk do you want to set up (gb)? ";
```

```

$vmdisk = <>;chomp $vmdisk;
print "\n";
print "Path to Save VM (ie. /vmfs/volumes/storage1/vm/";
$vmppath = <>;chomp $vmppath;
print "\n";

```

- Add new prompt in Defined Templates section, “option 2.”

```

$vmos = "winnetstandard";
print "Path to Save VM (ie. /vmfs/volumes/storage1/vm/";
$vmppath = <>;chomp $vmppath;
print "\n";

```

- Update subroutine “writevmx”.

```

$file = $vmppath . $vmname . ".vmx";

```

- Update subroutine “createdisk”.

```

$scr = "vmkfstools -c " . $vmdisk . "g " . $vmppath . $vmname . ".vmdk
-a lsilogic";
system("$scr");

```

- Update subroutine “registerVM”.

```

$rg = "vmware-cmd -s register " . $vmppath . $vmname . ".vmx";
system("$rg");

```

- Update subroutine “setper”.

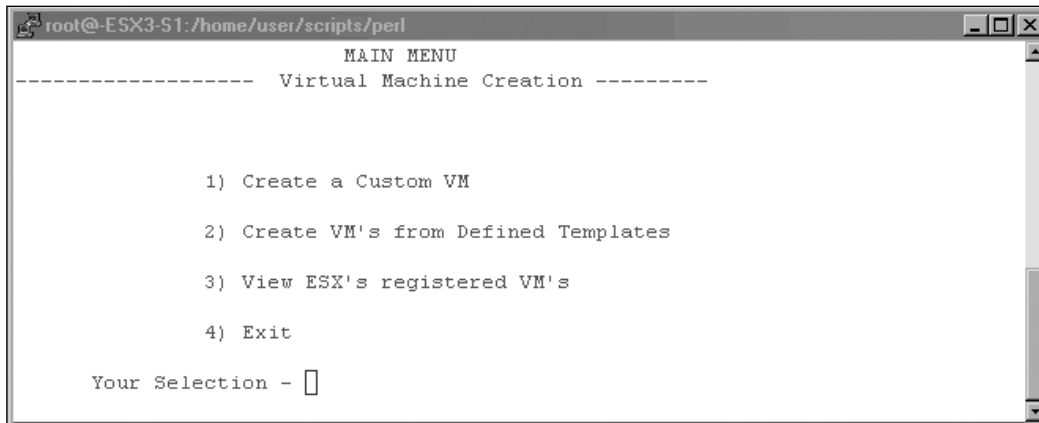
```

$pm = "chmod 755 " . $vmppath . $vmname . ".vmx";
system("$pm");

```

The script will now prompt you for VM destination when creating new VMs. Please note that when entering the destination file path, you should include the leading and trailing “/”.

When the script in Code Listing 4.7 is executed on the ESX server, a menu will be displayed (see Figure 4.14).

Figure 4.14 The Perl Script VM Creation Menu

Perl Script Components

Utilizing the script in Code Listing 4.7, you can do the following tasks:

- Create a custom VM with parameters that you supply.
- Create VMs from predefined templates.
- View listing of VMs currently registered on the ESX server.
- Exit the script.

This script was written to be easily customized by you, the reader. Variables have been set up for key VM-related options enabling simple modifications. Let's dissect this script to get a better understanding of VmPerl. Code Listing 4.8 shows the key variables.

Code Listing 4.8 Scripted Creation of VM with Perl Key Variables

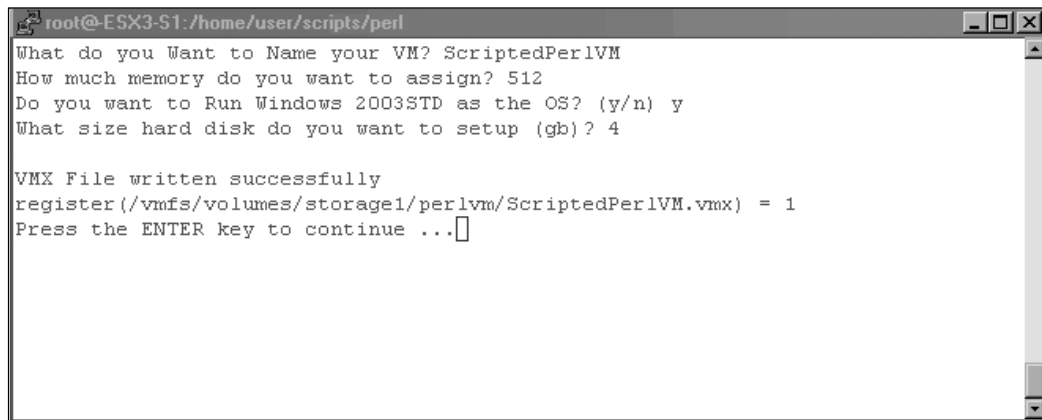
```

$vmname = virtual machine name, will be used for disk as well
$vmmem  = amount of memory assigned to VM
$vmos   = OS that VM is configured for
$vmdisk = size of VM disk
  
```

These variables in the program are either dynamic or static depending upon which option in the script you choose. The first option presented on the

menu shown in Figure 4.14 is **Create A Custom VM**. This option will prompt you for the variables listed in Code Listing 4.8, as shown in Figure 4.15.

Figure 4.15 Perl Script Custom Creation of VM



If you select the second option, **Create VM's from Defined Templates**, the values are set statically in that section. Code Listing 4.9 shows an example where these values are set in the code.

Code Listing 4.9 Perl Script Static Variables for Template VM Creation

```

if ($menuopt == 1) {
    $vmname = "2003std25m4gb";
    $vmmem = "256";
    $vmdisk = "4";
    $vmos = "winnetstandard";
}
  
```

It's a simple task to add additional menu options for more templates. Adding sections like those in Code Listing 4.9 will enable you to define a bigger selection of templates for your environment.

Master Craftsman...

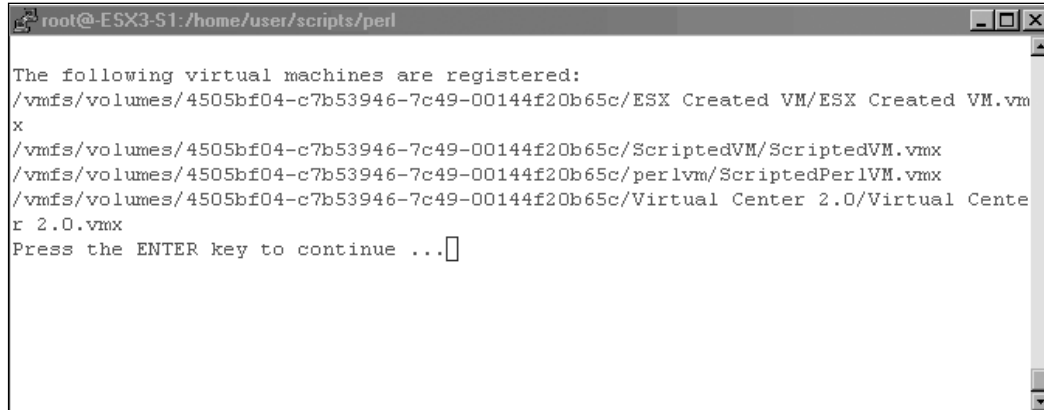
VM Procurement Automation

The example script provides basic VM procurement with two templates and a custom VM creation option. Typical production environments have a multitude of system types and requirements. You can modify the script to meet your needs and provide for procurement of VMs for varying situations such as:

- Procurement of new virtual machines for customers
- Procurement of groups of servers
- Automated environment setups (lab testing, and so on)

Because the script is written in VmPerl, which is just a VMware-customized version of Perl, you can leverage Perl's additional features and characteristics. Perl code can be executed from within a Web browser enabling you to create a VM procurement Web site. You could host this new Web site on the ESX server itself, and create a "Self-Service" procurement architecture. These are just some suggestions and ideas to get you thinking about the possibilities that VmPerl provides.

The third menu choice option in the script is View ESX's Registered VM's. This section utilizes the VmPerl API to access the ESX server. For more information on the VmPerl API, download the ESX server SDK. This section of the code connects to the local ESX server with your current userid and password on port 902. It then queries ESX Server for a listing of registered VMs. Figure 4.16 shows the output generated by this option.

Figure 4.16 Perl Script Query of ESX Server for Registered VMs

```
root@ESX3-S1:/home/user/scripts/perl
The following virtual machines are registered:
/vmfs/volumes/4505bf04-c7b53946-7c49-00144f20b65c/ESX Created VM/ESX Created VM.vmx
x
/vmfs/volumes/4505bf04-c7b53946-7c49-00144f20b65c/ScriptedVM/ScriptedVM.vmx
/vmfs/volumes/4505bf04-c7b53946-7c49-00144f20b65c/perlvm/ScriptedPerlVM.vmx
/vmfs/volumes/4505bf04-c7b53946-7c49-00144f20b65c/Virtual Center 2.0/Virtual Center 2.0.vmx
Press the ENTER key to continue ...
```

VmPerl Commands

VmPerl by itself cannot create virtual machine files or register virtual machines. To accomplish these tasks, we must use the tools available. The sample VmPerl script utilizes the command “system” to access the following VMware tools:

- *vmkfstools*
- *vmware-cmd*

Do those tools sound familiar? By now you’ve become quite adept at utilizing these tools for the creation of virtual machines. The latter sections of the code contain the subroutines that handle the virtual machine disk creation *createdisk*, and VM registration *registervm*. It is in these subroutines that we use the tools listed earlier.

Utilizing the example script, and with your working knowledge of the VMware tools from previous sections, you should have a competent understanding of how to create virtual machines from a VmPerl script.

Cloning Virtual Machines Utilizing ESX Shell Scripts

As we’ve seen in previous sections, VMware provides you with built-in tools to accomplish most virtualization tasks. Cloning is the process of copying an

172 Chapter 4 • Building a VM

existing virtual machine's virtual disk to a new file for a new virtual machine to use. The source virtual machine is considered to be the template VM. Understand that cloning creates an exact replica of the VM's disk contents. It is very similar to disk imaging. So when this clone is configured and turned on in ESX Server it will come up with all the same attributes as the template VM. To address this, the template VM should be prepared in advanced for cloning. For Microsoft Windows-based servers, you would use *Sysprep* to prepare the template image for cloning. Using template images saves an enormous amount of time when procuring servers for all types of situations. Because the template VM has the operating system preinstalled and configured, setup time for new systems is drastically reduced.

To clone an existing template virtual machine, we will use the VMware utility *vmkfstools*. We will use the *-i* option which instructs *vmkfstools* to import an existing template VM's disk file (VMDK) and copy it. The command syntax is as follows:

```
vmkfstools -I /pathToTemplateVM/template.vmdk  
/pathToDestinationVM/newvm.vmdk
```

The cloning process does not create a virtual machine configuration file or register the new virtual machine with ESX Server. We will leverage what you've learned in the previous sections to modify the ESX shell script from Code Listing 4.6.

We need to modify the part of the script that calls *vmkfstools* to create the 4GB virtual disk. We are instead going to use the *-i* command and clone an existing virtual disk. If you've been implementing the scripts in the previous sections, then you will have an example virtual machine disk that we can use for this section. If not, create a quick empty VM via any of the ESX GUI methods: Virtual Client, Virtual Infrastructure Client (ESX 3.0), Web, and so on.

Once you have your source template virtual disk ready, go ahead and edit the code to support cloning (see Code Listing 4.10).

Code Listing 4.10 ESX Shell Script VM Creation Utilizing Cloning

```
##### VM Creation Script Utilizing Cloning #####
#Script Version 1.2
#Author David E. Hart
#Date 10-05-06
#
#-----+
# Purpose|
#-----+-----
# This script will create a VM utilizing the cloning option of # the
vmkfstools command tool;
# The New Virtual Machine Configuration will be set as follows
# Virtual Machine Name = ScriptedCloneVM
# Location of Virtual Machine = /VMFS/volumes/storage1/ScriptedVM
# Virtual Machine Type = "Microsoft Windows 2003 Standard"
# Virtual Machine Memory Allocation = 256 meg
#
#-----+
#Custom Variable Section for Modification|
#-----+-----
#NVM is name of virtual machine(NVM). No Spaces allowed in name
#NVMDIR is the directory which holds all the VM files
#NVMOS specifies VM Operating System
#-----+
#####

### Default Variable settings - change this to your preferences
NVM="ScriptedCloneVM"      # Name of Virtual Machine
NVMDIR="ScriptedCloneVM"  # Specify only the folder name to be created;
NOT the complete path
NVMOS="winnetstandard"    # Type of OS for Virtual Machine
VMMEMSIZE="256"           # Default Memory Size

### End Variable Declaration
```

174 Chapter 4 • Building a VM

```
mkdir /vmfs/volumes/storage1/$NVMDIR # Creates directory
exec 6>&1                               # Sets up write to file
exec 1>/vmfs/volumes/storage1/$NVMDIR/$NVM.vmx # Open file
# write the configuration
echo config.version = "'6'"           # For ESX 3.x the value is 8
echo virtualHW.version = "'3'"       # For ESX 3.x the value is 4
echo memsize = "'$VMMEMSIZE'"
echo floppy0.present = "'TRUE'"      # setup VM with floppy
echo displayName = "'$NVM'"          # name of virtual machine
echo guestOS = "'$NVMOS'"
echo
echo ide0:0.present = "'TRUE'"
echo ide0:0.deviceType = "'cdrom-raw'"
echo ide0:0.startConnected = "'false'" # CDROM enabled
echo floppy0.startConnected = "'FALSE'"
echo floppy0.fileName = "'/dev/fd0'"
echo Ethernet0.present = "'TRUE'"
echo Ethernet0.networkName = "'VM Network'" # Default network
echo Ethernet0.addressType = "'vpx'"
echo
echo scsi0.present = "'true'"
echo scsi0.sharedBus = "'none'"
echo scsi0.virtualDev = "'lsilogic'"
echo scsi0:0.present = "'true'"      # Virtual Disk Settings
echo scsi0:0.fileName = "'$NVM.vmdk'"
echo scsi0:0.deviceType = "'scsi-hardDisk'"

echo
# close file
exec 1>&-

# make stdout a copy of FD 6 (reset stdout), and close FD6
```

```
exec 1>&6
exec 6>&-

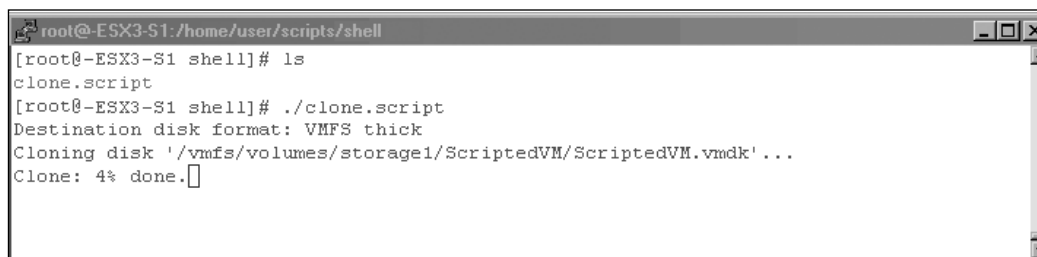
# Change permissions on the file so it can be executed by anyone
chmod 755 /vmfs/volumes/storage1/$NVMDIR/$NVM.vmx

#Clone existing Template VM's VMDK into current directory
cd /vmfs/volumes/storage1/$NVMDIR #change to the VM dir
vmkfstools -i /vmfs/volumes/storage1/ScriptedVM/ScriptedVM.vmdk $NVM.vmdk

#Register VM
vmware-cmd -s register /vmfs/volumes/storage1/$NVMDIR/$NVM.vmx
```

When you execute the script, the status of the cloning process will be displayed (see Figure 4.17).

Figure 4.17 Cloning Process



```
root@-ESX3-S1:/home/user/scripts/shell
[root@-ESX3-S1 shell]# ls
clone.script
[root@-ESX3-S1 shell]# ./clone.script
Destination disk format: VMFS thick
Cloning disk '/vmfs/volumes/storage1/ScriptedVM/ScriptedVM.vmdk'...
Clone: 4% done.█
```

When the script finishes, you will have a new cloned copy of the template VM ready for use. Log on to the ESX GUI and validate that the new VM is registered and available.

The ability to script the cloning of existing template VMs allows you to pre-stage your virtual environments for your particular needs. For instance, you could have a Windows Lab of four servers pre-staged. Just run the WindowsLab script and all four VMs are created and ready to go. In the next chapter, you will learn how to perform operations on VMs such as starting and stopping VMs via scripting.

Cloning Virtual Machines Utilizing VmPerl Scripts

You already know the benefits of cloning, but by utilizing the VmPerl scripting language you can build scripted procurement systems. VmPerl provides you more flexibility and more functionality than shell scripting. This allows you to be more creative in your approach to VM creation. We will add the cloning functionality to the example script in Code Listing 4.7. We will also add a new menu option for cloning and a new subroutine. In addition, we will use the VMware command tool *vmkfstools* with the *-i* option for cloning as we did in the previous chapter. Code Listing 4.11 shows the new Perl script with cloning.

Code Listing 4.11 Scripted VM Creation with Perl Utilizing Cloning

```
#!/usr/bin/perl -w

use VMware::VmPerl;
use VMware::VmPerl::Server;
use VMware::VmPerl::ConnectParams;
#use strict;

##### VM Menu Driven Creation Script with Cloning #####
#Script Version 1.3
#Author David E. Hart
#Date 10-05-06
#
#-----+
#Purpose |
#-----+
# This script presents a menu for automatically building
# virtual machine config files (VMX) and disk files (VMDK)
# This script demonstrates how to automate the setup
# of virtual environments and includes cloning of VMs
#-----+

```

```

#Custom Variables Section |
#-----+
#vmname = virtual machine name, will be used for disk as well
#vmmem = amount of memory assigned to VM
#vmos = OS that VM is configured for
#vmdisk = size of VM disk
#####

main:      # main menu

system("clear");
print "                MAIN MENU \n";
print "----- Virtual Machine Creation ----- \n";
print "\n";
print "\n";
print "\n";
print "                1) Create a Custom VM \n";
print "\n";
print "                2) Create VM's from Defined Templates \n";
print "\n";
print "                3) View ESX's registered VM's \n";
print "\n";
print "                4) Clone an Existing VM \n";
print "\n";
print "                5) Exit \n";
print "\n";
print "        Your Selection - ";
$menuopt = <>; chomp $menuopt;      # Get user selection
if ($menuopt == 1) { # Get input for custom VM
    system("clear");
    print "What do you Want to Name your VM? ";
    $vmname = <>; chomp $vmname; # use chomp to remove carriage return
    print "How much memory do you want to assign? ";
    $vmmem = <>;chomp $vmmem;

```

178 Chapter 4 • Building a VM

```
print "Do you want to run Windows 2003STD as the OS? (y/n) ";
$vmos = <>;chomp $vmos;
if ($vmos eq "y") {
    $vmos = "winNetStandard";
}          # Only 2 options for this example
else {
    print "Do you want to run Windows 2003Ent as the OS? (y/n) ";
    $vmos2 = <>;chomp $vmos2;
    if ($vmos2 eq "y") {
        $vmos = "winnetenterprise";
    }
}

print "What size hard disk do you want to set up (gb)? ";
$vmdisk = <>;chomp $vmdisk;
print "\n";
$x = writevmx();    # Subrouting for creating VMX file
if ($x == 1) {
    print "VMX file written successfully \n";
}

$w = setper();      # Subroutine to set permissions so anyone can
use VM
if ($w == 1) {
    print "Permissions set successfully \n";
}

$y = createdisk(); # subrouting to create VMDK disk file
if ($y == 1) {
    print "Virtual disk created successfully \n";
}

$z = registervm(); # subroutine to register VM with ESX
if ($z == 1) {
    print "VM registered successfully \n";
}

print "Press the ENTER key to continue ...";
$pause = <STDIN>;
```

```
goto main

}

if ($menuopt == 2) { # option to display the templates
menu1:
system("clear");
print "                Defined Templates \n";
print "                ----- \n";
print "\n";
print "\n";
print "                1) Windows 2003std VM with 256m, 4gb drive \n";
print "\n";
print "                2) Windows 2003ent VM with 1gig, 8gb drive \n";
print "\n";
print "\n";
print "\n";
print "\n";
print "    Your Selection - ";
$menuopt = <>; chomp $menuopt;
if ($menuopt == 1) {
    $vmname = "2003std25m4gb";
    $vmmem = "256"; # change and add on similar sections
    $vmdisk = "4"; # to create templates for your environment
    $vmos = "winnetstandard";
    $x = writevmx();
    if ($x == 1) {
        print "VMX file written successfully \na";
    }
    $w = setper();
    if ($w == 1) {
        print "Permissions set successfully \na";
    }
    $y = createdisk(); # Call subroutines to create VM's
    if ($y == 1) {
```

180 Chapter 4 • Building a VM

```
        print "Virtual disk created successfully \na";
    }
    $z = registervm();
    if ($z == 1) {
        print "VM registered successfully \na";
    }
    print "Press the ENTER key to continue ...";
    $pause = <STDIN>;
    goto main
}
if ($menuopt == 2) {
    $vmname = "2003Ent1gb8gb";
    $vmmem = "1024";
    $vmdisk = "8";
    $vmos = "winnetenterprise";
    $x = writevmx();
    if ($x == 1) {
        print "VMX file written successfully \na";
    }
    $w = setper();
    if ($w == 1) {
        print "Permissions set successfully \na";
    }
    $y = createdisk();
    if ($y == 1) {
        print "Virtual disk created successfully \na";
    }
    $z = registervm();
    if ($z == 1) {
        print "VM registered successfully \na";
    }
    print "Press the ENTER key to continue ...";
    $pause = <STDIN>;
    goto main
```

```
    }
    else {
        goto menu1;
    }

}

if ($menuopt == 3) {    # Use a function of VmPerl to display registered VMs
    system("clear");
    my ($server_name, $user, $passwd) = @ARGV; # Assume running in ESX
server
    my $port = 902;                                # with appropriate
rights

VMware::VmPerl::ConnectParams::new($server_name,$port,$user,$passwd);
    VMware::VmPerl::ConnectParams::new(undef,$port,$user,$passwd);
    my $connect_params = VMware::VmPerl::ConnectParams::new();

    # Establish a persistent connection with server
    my $server = VMware::VmPerl::Server::new();
    if (!$server->connect($connect_params)) {
        my ($error_number, $error_string) = $server->get_last_error();
        die "Could not connect to server: Error $error_number:
$error_string\n";
    }

    print "\n\nThe following virtual machines are registered:\n\n";

    # Obtain a list containing every config file path registered with the
server.
    my @list = $server->registered_vm_names();
    if (!defined($list[0])) {
        my ($error_number, $error_string) = $server->get_last_error();
        die "Could not get list of VMs from server: Error $error_number:
".
```

182 Chapter 4 • Building a VM

```

        "$error_string\n";
    }

    print "$_\n" foreach (@list);

    # Destroys the server object, thus disconnecting from the server.
    undef $server;
    print "Press the ENTER key to continue ...";
    $pause = <STDIN>;
    goto main
}

if ($menuopt == 4) {
    system("clear");
    print "                Clone Existing VM.s \n";
    print "                ----- \n";
    print "\n";
    print "\n";
    print "        1) Clone ScriptedVM \n";
    print "\n";
    print "        2) Clone ScriptedPerlVM \n";
    print "\n";
    print "\n";
    print "\n";
    print "\n";
    print "    Your Selection - ";
    $menu4opt = <>; chomp $menu4opt;
    if ($menu4opt == 1) {
        $vmname = "ScriptedPerlCloneVM";
        $vmmem = "256"; # change and add on similar sections
        $vmdisk = "4"; # to create templates for your environment
        $vmos = "winnetstandard";
        $vmppath = "/vmfs/volumes/storage1/ScriptedVM/ScriptedVM.vmdk";
        $x = writevmx();
    }
}

```

```
    if ($x == 1) {
    print "VMX file written successfully \na";
    }
    $w = setper();
    if ($w == 1) {
    print "Permissions set successfully \na";
    }
    $y = clonedisk();          # Call subroutines to create VM's
    if ($y == 1) {
    print "Virtual disk cloned successfully \na";
    }
    $z = registervm();
    if ($z == 1) {
    print "VM registered successfully \na";
    }
    print "Press the ENTER key to continue ...";
    $pause = <STDIN>;
    goto main
}
if ($menu4opt == 2) {
    $vmname = "ScriptedPerlVMClone";
    $vmmem = "1024";
    $vmdisk = "8";
    $vmos = "winnetenterprise";
    $vmpath = "/vmfs/volumes/storage1/perlvm/ScriptedPerlVM";

    $x = writevmx();
    if ($x == 1) {
    print "VMX file written successfully \na";
    }
    $w = setper();
    if ($w == 1) {
    print "Permissions set successfully \na";
    }
}
```

184 Chapter 4 • Building a VM

```

        $y = clonedisk();
        if ($y == 1) {
            print "Virtual disk cloned successfully \na";
        }
        $z = registervm();
        if ($z == 1) {
            print "VM registered successfully \na";
        }
        print "Press the ENTER key to continue ...";
        $pause = <STDIN>;
        goto main
    }
else {
    goto menu1;
}

}

if ($menuopt == 5) {
    goto endl
}

sub writevmx {          # Subroutine to create VM's VMX config file

#       $file = '/vmfs/volumes/storage1/perlvm/perlvm.vmx';          #
Name the file
    $file = "/vmfs/volumes/storage1/perlvm/" . $vmname . ".vmx";
    open(INFO, ">$file");    # Open for output
    print INFO 'config.version = "6" ' . "\n";
    print INFO 'virtualHW.version = "3" ' . "\n";
    print INFO 'memsize = "' . $vmmem . "' ' . "\n";
    print INFO 'floppy0.present = "TRUE" ' . "\n";
    print INFO 'displayName = "' . $vmname . "' ' . "\n";

```

```

print INFO 'guestOS = ' . $vmos . ' ' . "\n";
print INFO 'ide0:0.present = "TRUE" ' . "\n";
print INFO 'ide0:0.deviceType = "cdrom-raw" ' . "\n";
print INFO 'ide:0.startConnected = "false" ' . "\n";
print INFO 'floppy0.startConnected = "FALSE" ' . "\n";
print INFO 'floppy0.fileName = "/dev/fd0" ' . "\n";
print INFO 'Ethernet0.present = "TRUE" ' . "\n";
print INFO 'Ethernet0.connectionType = "monitor_dev" ' . "\n";
print INFO 'Ethernet0.networkName = "VM Network" ' . "\n";
print INFO 'Ethernet0.addressType = "vpx" ' . "\n";
print INFO 'scsi0.present = "true" ' . "\n";
print INFO 'scsi0.sharedBus = "none" ' . "\n";
print INFO 'scsi0.virtualDev = "lsilogic" ' . "\n";
print INFO 'scsi0:0.present = "true" ' . "\n";
print INFO 'scsi0:0.fileName = ' . $vmname . '.vmdk' ' . "\n";
print INFO 'scsi0:0.deviceType = "scsi-hardDisk" ' . "\n";

close(INFO); # Close the file
}

sub createdisk { # Subroutine to create virtual disk
    $scr = "vmkfstools -c " . $vmdisk . "g " . "
/vmfs/volumes/storage1/perlvm/" . $vmname . ".vmdk -a lsilogic";
    system("$scr");
};

sub clonedisk { # Subroutine to create virtual disk
    $scr = "vmkfstools -i " . $vmpath . " " . "
/vmfs/volumes/storage1/perlvm/" . $vmname . ".vmdk";
    system("$scr");
};

sub registervm { # Subroutine to register VM with ESX server
    $rg = "vmware-cmd -s register /vmfs/volumes/storage1/perlvm/" .
$vmname . ".vmx";

```

186 Chapter 4 • Building a VM

```
        system("$rg");
    }

sub setper{          # Subroutine to set permission on VMX file
    $pm = "chmod 755 /vmfs/volumes/storage1/perlvm/" . $vmname .
    ".vmx";
    system("$pm");
}

endl:
```

The preceding code is highlighted with the changes necessary to support cloning. When the code is executed, you now have a new menu option #4, for cloning of virtual machines. The code currently clones ScriptedVM.vmdk and ScriptedPerlVM.vmdk, created from previous sections. You can easily modify the code to request the name of the VMs to clone. You could even have the code generate a list of VMs registered with the ESX server and then you would select from this list. The script is provided as an example of how you would go about setting up your own VMs to use as templates and how to automate creating clones of these. Go ahead and expand the sample script to include other options such as “lab setup” where the option clones a series of virtual templates to set up a virtual test environment.

Master Craftsman...

Using Clones to Set Up Virtual Environments

The example script provides basic VM procurement via custom entry, templates, and cloning. Custom VM entry and templates provide you new VMs ready for the installation of an operating system, while cloning provides you with a prebuilt virtual machine ready for use. Create clone templates of your most common server types in your environment for fast deployment in your virtual infrastructure. You can, in essence, set up virtual labs in a matter of minutes versus hours.

Summary

In this chapter, you learned how to use the built-in command-line tools from VMware—namely, *vmkfstools* and *vmware-cmd*—to build and clone virtual machines. You also learned how to use ESX shell scripting to incorporate these tools and automate the VM and cloning process. We showed you how to employ VmPerl for advanced scripting of VM creation and cloning. We then showed you how to use the code examples to build a rough VM creation and cloning architecture for you to expand on. You should now have a good understanding of what you can script on the ESX server as it relates to virtual machine creation.

