Realtime publishers

Automating Windows 7 Installation for Desktop and VDI Environments

Greg Shields

Copyright Statement

Chapter 8: Integrating Automated Windows 7 Installation into VDI Environments

When is a desktop not really a desktop? *When it's a virtual desktop, of course!*

From the user's perspective, a virtual instance of Windows 7 might look the same as a physical instance. But you're an IT professional. You're aware of all the extra forces working behind the scenes that are required to make that virtual instance happen. Virtualization, presentation, transport protocols, automated deployment, and snapshotting all comprise the virtual desktop administrator's toolkit for deploying even the most basic Windows 7 instance.

Complicating this process even more is the multitude of desktop virtualization solutions on the market today. VMware View, Citrix XenDesktop, and Microsoft Remote Desktop Virtualization Host represent three of the major players. They're not the only ones. "Smaller" alternatives like Quest, Wanova, MokaFive, NComputing, and others now release their own solutions that create a Virtual Desktop Infrastructure or VDI.

When it comes to Windows 7 itself, however, there is good news. What you've learned so far in this guide directly assists in creating virtual machines inside a VDI infrastructure, notwithstanding which vendor writes the software. No matter whose technology connects your users to VMs, the processes you now know for configuring Windows remain relatively the same. That knowledge makes my life easier as this book's author. It also makes your life easier if you've been following along throughout its pages.

That's why this chapter won't delve too deeply into the specifics behind each VDI platform. The exact clicks required to build a desktop template in Citrix XenDesktop are quite different than those inside VMware View. That same series of clicks will surely change *within each product* as they evolve over time. So, I'll leave the specifics to the vendor documentation.

Rather than focusing on the VDI products, this final chapter focuses on Windows itself. The reason? Windows 7, as you know, is designed to be an "everything for everyone" operating system (OS). Far from a single-purpose OS, Windows 7 can just as easily be a home desktop for one individual as a virtual business desktop for another. The difference is in the tuning.

In this chapter, I'll share with you the high-level process you'll use in preparing Windows 7 for VDI deployment. As you can imagine, many of these tuning suggestions remove the "extra" bits of Windows that contribute to its resource use. With VDI, many copies of Windows operate simultaneously on the same server. Thus, the task of eliminating unnecessary services and processes must be done to guarantee overall performance.

Finally, going down the virtual route adds the potential for a big win. Namely, *VDI rewards smart administration*. The smarter you are in tuning your Windows instances, the better they will perform in aggregate. The more resources you conserve inside each individual desktop leaves more available for others. Smarter administration in VDI means squeezing more desktops on fewer servers while still preserving each user's experience. In the end, with greater density, you'll get more out of less.

Step Fifteen: Integrating MDT into VDI Deployment

With that introduction, let's step back into Windows 7 deployment, but this time focus on virtual deployment. We begin by bouncing back to our old friend the MDT. You might at this point be thinking, "Wait a minute. We're moving backwards from ConfigMgr to MDT?" Absolutely. Although the extra functionality delivered by ConfigMgr was necessary in the previous chapter's zero-touch installation scenario, you can use MDT (at no cost!) as your customization platform for VDI deployment.

One of VDI's biggest differences is in how the resulting WIM image ultimately gets deployed. Thus far, the WIM files we've created were designed to be deployed to physical desktops. Deploying to virtual desktops uses a different set of steps. Replacing the physical desktop's hard disk is a virtual hard disk. For VMware VDI, that virtual disk file is a VMDK. Microsoft and Citrix use a VHD file. Combined with that VMDK or VHD file are others that describe the configuration of the virtual machine itself, such as its memory configuration or number of processors. All are necessary to describe the characteristics of the virtual machine.

Creating a Virtual Machine and Deploying an OS

I mentioned earlier that different VDI platforms use different mechanisms for managing their virtual machines. This chapter will attempt to stay as product-neutral as possible, so I'll keep things simple by sticking with Microsoft's Hyper-V R2 platform. If you're familiar with how to create and work with virtual machines, you should be able to easily translate what you see here into your platform of choice.

Creating a new virtual machine with Hyper-V starts in the Hyper-V Manager by clicking New | Virtual Machine (see Figure 8.1). In the resulting wizard, follow through its steps to create a new virtual machine somewhere on the Hyper-V server. Although most VDI deployments leverage SANs for production deployments, our interest at this point is only in learning to deploy a WIM file's contents to a virtual disk.

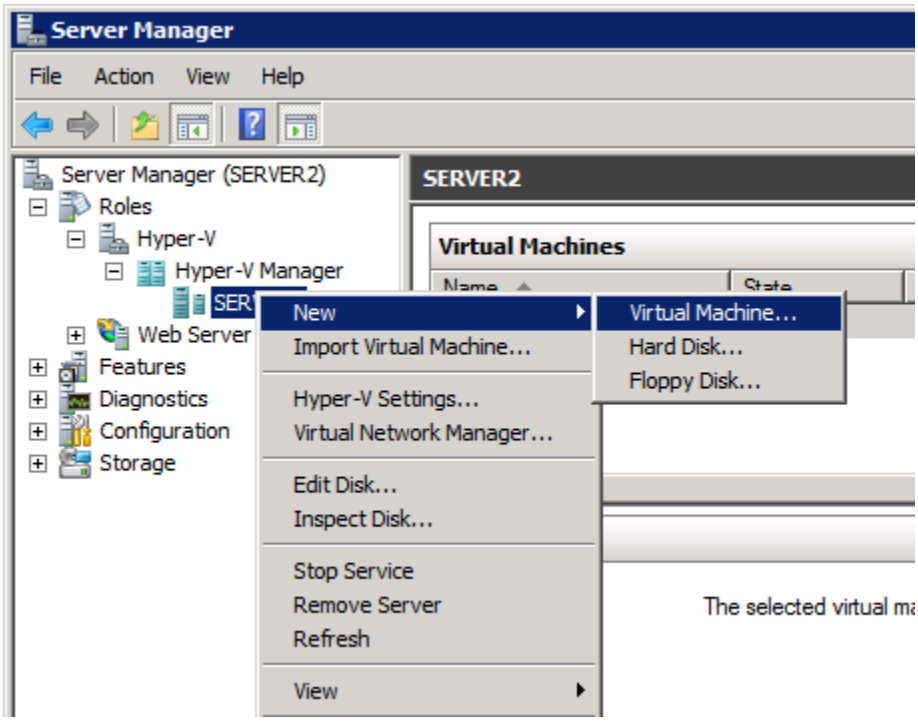

Figure 8.1: Creating a new virtual machine.

Windows 7 virtual machines tend not to require as much RAM as is commonly installed to physical machines. Although Windows 7 can operate in as little as 512MB of RAM, real-world best practices suggest a starting point of 1536MB of RAM. One great fact about virtual machines is that you can always adjust that quantity up or down later if you see performance problems or unused resources.

Connect the virtual machine to an existing virtual network and create a virtual hard disk when prompted. The final screen in the wizard (see Figure 8.2) discusses the options available for deploying an OS to this newly-created virtual machine. Take a look at the radio button at the bottom of the screen that you may not have noticed before. That radio button enables you to install an OS from a network-based installation server.

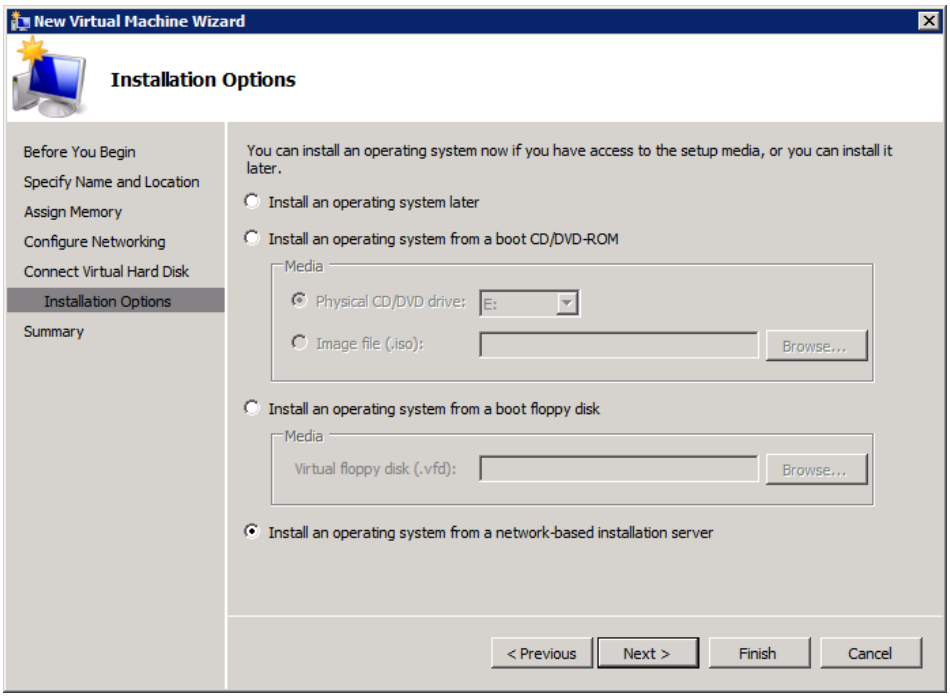

Figure 8.2: Installation options.

Recall that our entire series of activities to this point has created the very network-based installation server this radio button references. By selecting this radio button, you are effectively PXE booting the virtual desktop and connecting it to your WDS server to receive an OS image.

Before booting the virtual machine, however, right-click the powered-down computer and view its settings. Look for its BIOS settings, similar to what you see in Figure 8.3. You'll see that the virtual machine has been configured to boot from its legacy network adapter. This legacy network adapter is a special network adapter that's used by Hyper-V just to get the machine started during its initial configuration. Being "legacy," it is by no means high performance; however, it does include the necessary code to integrate with your WDS server.

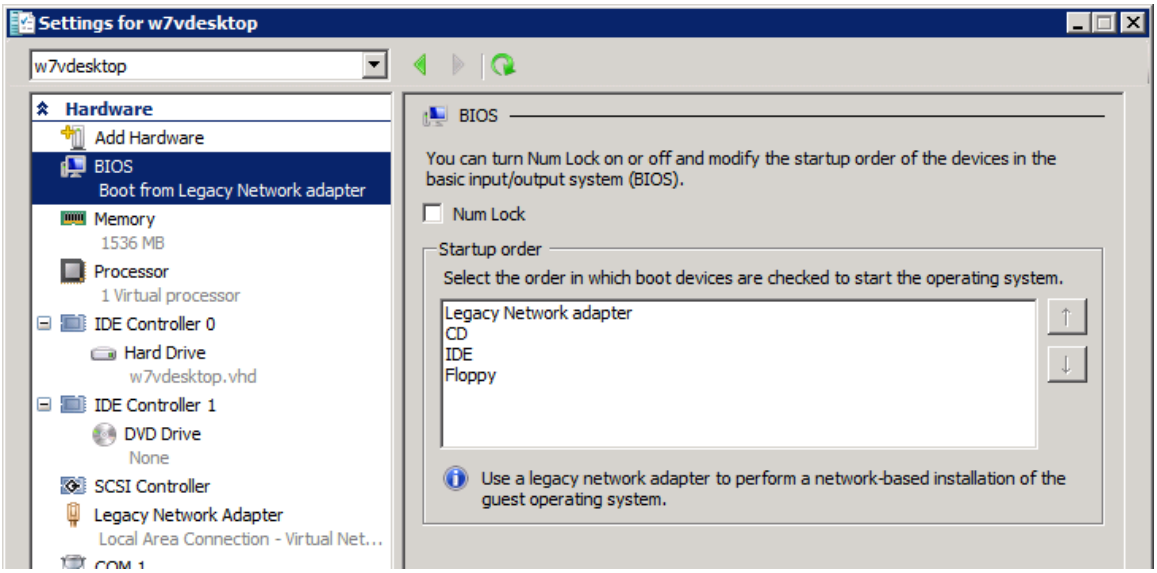

Figure 8.3: Boot from legacy network adapter.

Note

Microsoft recommends the use of the network adapter and not the legacy network adapter when a machine (server or desktop) is in production. The reason is that the network adapter enjoys a set of performance and other optimizations not available on the legacy network adapter. You'll eventually switch to the "regular" adapter after installing Hyper-V's Integration Components.

Other hypervisors use a similar series of virtual adapters and drivers. For example, VMware View requires virtual machines to use VMware's e1000 network card driver to connect to a WDS server, but suggests later switching to their VMXNET3 driver once provisioned.

Connect to the virtual machine, and power it on. You should quickly spot a familiar screen (see Figure 8.4) if the Hyper-V host's virtual network successfully communicates with the WDS server. In that screen, you'll see that the virtual machine is ready for a network service boot. Hitting the F12 button at this point will connect the virtual machine to the WDS server for an OS deployment.

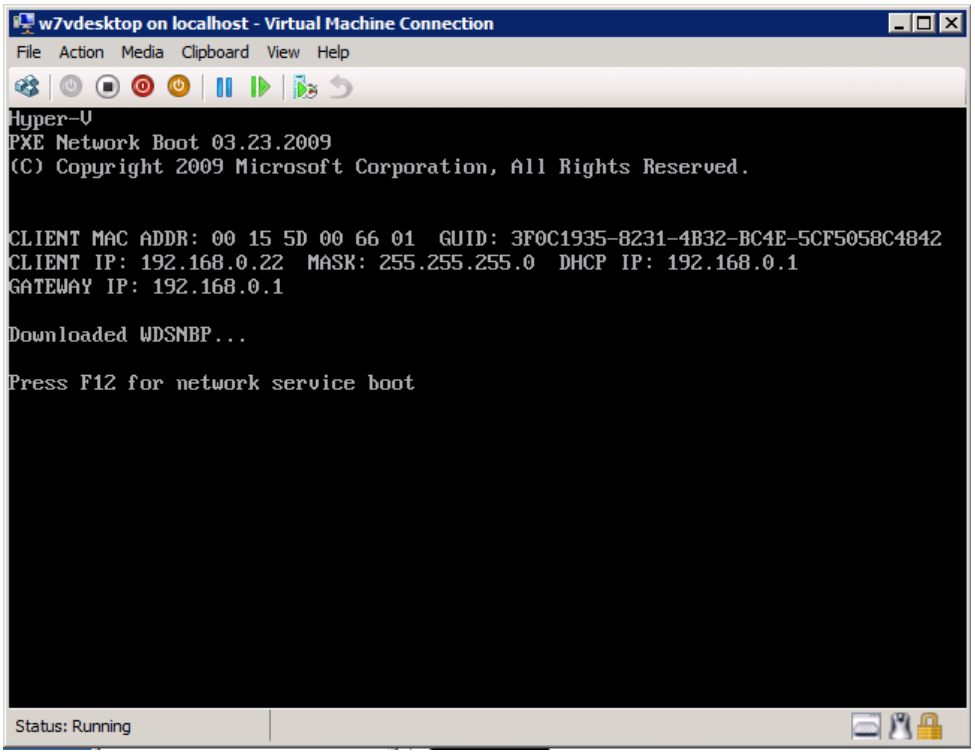

Figure 8.4: PXE booting the virtual machine.

Injecting Drivers and Virtual Tools

Voila! You're successfully deploying a Windows 7 image to a virtual machine! Not much to it, eh? In reality, this book's previous chapters have led you through most of the work in automating this deployment process. If you do nothing else at this point, you have, at the very least, created your first Windows 7 virtual machine atop Hyper-V.

Yet, as you know, there's always more to the story. You will need to add Hyper-V's drivers and virtual machine tools to your MDT driver store. You've done this before with the VMware Workstation drivers in Chapter 2. Hyper-V's drivers and tools are called the Hyper-V Integration Components and are located on any Hyper-V server in the ISO file at C:\Windows\System32\vmguest.ISO. Mount this ISO to a drive letter to get at its contents, then unpack the drivers from their installation files within. Add them to MDT's driver store once you've gained access to the drivers themselves.

> **Note**
>
> Another common tactic to install a VDI platform's virtual tools to desktops uses an MDT application. You learned how to do this in Chapter 4 by creating and deploying a silenced installation during the OS deployment process. The only difference here is that the application being deployed is a collection of drivers (among other software). Using an MDT application replaces some of the need for driver injection by directly automating the installation of the VDI platform's virtual tools.

Tuning the Virtual Machine

With drivers ready to go, your next step will be to tune Windows 7 for use with VDI. These tuning activities typically involve shutting down unnecessary services and processes with the goal of streamlining Windows for best performance as a virtual machine.

In the early days of VDI, this process involved quite a bit of guess-and-check work. Today, however, most VDI vendors have developed a set of guidance documents to assist. Although each vendor releases its own documentation, you'll find a fairly significant overlap in what they suggest you should do. I'll introduce you to three of these documents that are available as of the time of this writing, then summarize some of what they suggest in what remains of this chapter. Those three documents are:

- *VMware View Optimization Guide for Windows 7*
- *Deploying Microsoft Windows 7 Virtual Desktops with VMware View*
- *Citrix Windows 7 Optimization Guide for Desktop Virtualization*

> **Note**
> Tuning guidance is always a moving target as our industry learns new tips and tricks. Thus, I am purposely not providing direct links to these documents, as they will evolve over time and potentially change locations. You should be able to search for their titles using your favorite Web browser.

Guidance for Services

Your first step in tuning Windows 7 starts by disabling the services that don't make sense within a VDI deployment. Unlike Windows XP, where nearly every service was enabled by default, Windows 7 does a better job of turning on only those services that it absolutely needs. That said, the default installation of Windows 7 still enables services that can be safely disabled when they aren't absolutely necessary.

Disabling these services also stops any associated processing, which has an effect on overall VDI environment performance. Collected from the three documents listed earlier, consider the following services as a starting list for those you might want to disable on your VDI virtual machines:

- **Background Intelligent Transfer Service (BITS)**. Some applications and services require BITS for functionality; test before removing
- **BitLocker Drive Encryption Service**. BitLocker is not recommended for use within many VDI architectures
- **Block Level Backup Engine Service, Microsoft Software Shadow Copy Provider, System Restore, Volume Shadow Copy Service, Windows Backup.** Used for protecting local data, which is unnecessary in a layered environment where data is stored off the desktop and issues are most often resolved by recreating the desktop
- **Desktop Window Manager Session Manager.** Enables Windows Aero, which will improve performance if disabled

- **Disk Defragmenter.** Disk defragmentation will have a significant impact on performance; pooled desktops are typically destroyed and re-created often enough that fragmentation does not create significant performance issues
- **Diagnostic Policy Service, Windows Error Reporting Service.** Used for troubleshooting and problem detection; less useful in a VDI environment where a common solution is to destroy and re-create the provisioned desktop
- **Function Discovery Resource Publication.** Publishes computer information on the network, which may not be necessary for enterprise deployments
- **Home Group Listener, Home Group Provider.** Home groups are unnecessary in enterprise deployments
- **Indexing Service.** Used for indexing local data, which is unnecessary in a layered environment where data is stored off the desktop
- **IP Helper.** Disable when IPv6 is not used
- **Microsoft iSCSI Initiator Service.** VDI desktops typically do not use local iSCSI disks
- **Offline Files.** Used for storing copies of network files, which is a use case not found in most online VDI deployments
- **Secure Socket Tunneling Protocol Service.** Used for VPN connectivity, which is not commonly a part of a VDI architecture
- **Security Center.** This service notifies users when security-related configurations are modified, which may be unnecessary in a fully-managed VDI architecture
- **SSDP Discovery, UPnP Host Service.** UPnP devices are not commonly attached to VDI virtual machines
- **Superfetch.** Used for caching commonly-used applications; less useful in a VDI environment where provisioned desktops are routinely destroyed and re-created
- **Tablet PC Input Service.** Tablet PCs are not commonly attached to VDI virtual machines
- **Themes.** This service adds personalization to the desktop experience but at the cost of added resource utilization
- **Windows Defender, Windows Firewall.** Some environments elect not to install anti-malware or host-based firewall software to VDI desktops due to their extremely short lifespan
- **Windows Media Center Receiver Service, Windows Media Center Scheduler Service, Windows Media Player Sharing Service.** Windows Media Center and Windows Media Player are not commonly used on VDI virtual machines
- **Windows Search.** Used in searching for data, which is unnecessary in a layered environment where data is stored off the desktop

- **Windows Update.** Virtual machine images are typically based on clones of a core image; when updates are ready for deployment, typically the core image is updated with clones regenerated thereafter; updates are typically not installed directly to cloned images
- **WLAN AutoConfig, WWAN AutoConfig.** Used for wireless LAN and mobile broadband configuration; not a common configuration with VDI virtual machines

Disabling these services can be accomplished via many mechanisms. Both Group Policy and Local Policies are two mechanisms to control configuration while desktops are in operation. Services can also be configured during OS installation by adding custom tasks to an MDT task sequence. These custom tasks, in this case the Run Command Line task, configures services by invoking the Windows *sc* command with appropriate options.

Service configuration obviously requires a functioning OS, thus they are typically disabled late in the task sequence after the core installation is complete. Figure 8.5 shows how the Run Command Line task might be added to a standard client task sequence in the MDT. There, the task has been added to the Custom Tasks step, which itself is a component of the State Restore step, by selecting Add | General | Run Command Line and entering the following text into the *Command line* field:

```
sc config wuauserv start= disabled
```

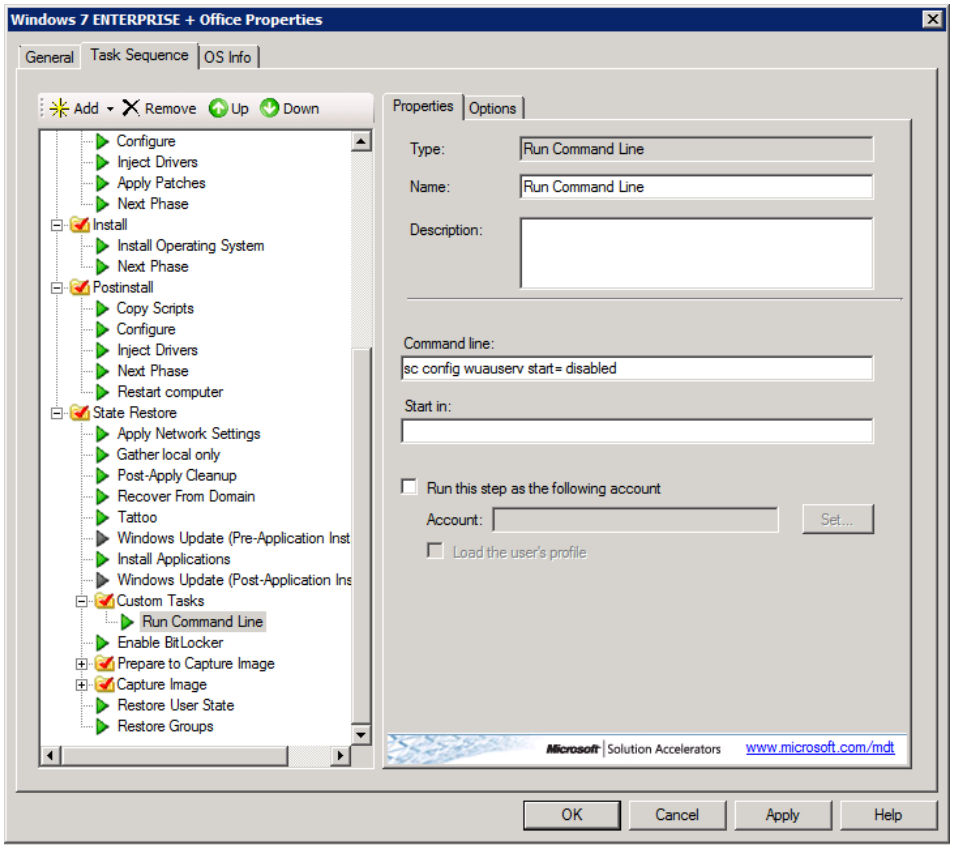

Figure 8.5: A custom task to disable a service.

The contents of the *Command line* field in Figure 8.5 show that the sc command has been configured to set the startup value to disabled for the wuauserv (Windows Update) service. Similarly-structured command lines can be used to modify the configuration of other services.

Guidance for Additional Configurations and Scripting

Services aren't the only configurations that require tuning. The guides noted earlier suggest other configurations in which resource-intensive activities such as full window dragging, font smoothing, and cursor blink are disabled. These other configurations can be set through registry manipulations, the use of native commands, or even Group Policy or Local Policies. Often, Loopback Policy Processing is used to apply user-specific settings to organizational units (OUs) that contain the VDI desktops' computer objects.

In the interests of space, I'll direct you to each document rather than reprinting their guidance here. That said, implementing some of these configurations at the time of installation is another area where the MDT's task sequences can assist. One way to invoke a series of changes all at once is through the use of scripts.

Recall that the MDT's Run Command Line task can invoke any executable or script that is natively supported by Windows 7. This support means that batch files, VBScripts, and (with a little extra effort) even PowerShell scripts can be used to make configuration changes.

Figure 8.6 shows a second Run Command Line task that's been added to the Custom Scripts step. That task runs a custom VBScript script named DisableCursorBlink.wsf by invoking the cscript.exe script host. The contents of the VBScript itself are unimportant for this discussion. As long as those contents correctly accomplish the task without generating an error or prompting the user, the script will execute correctly.

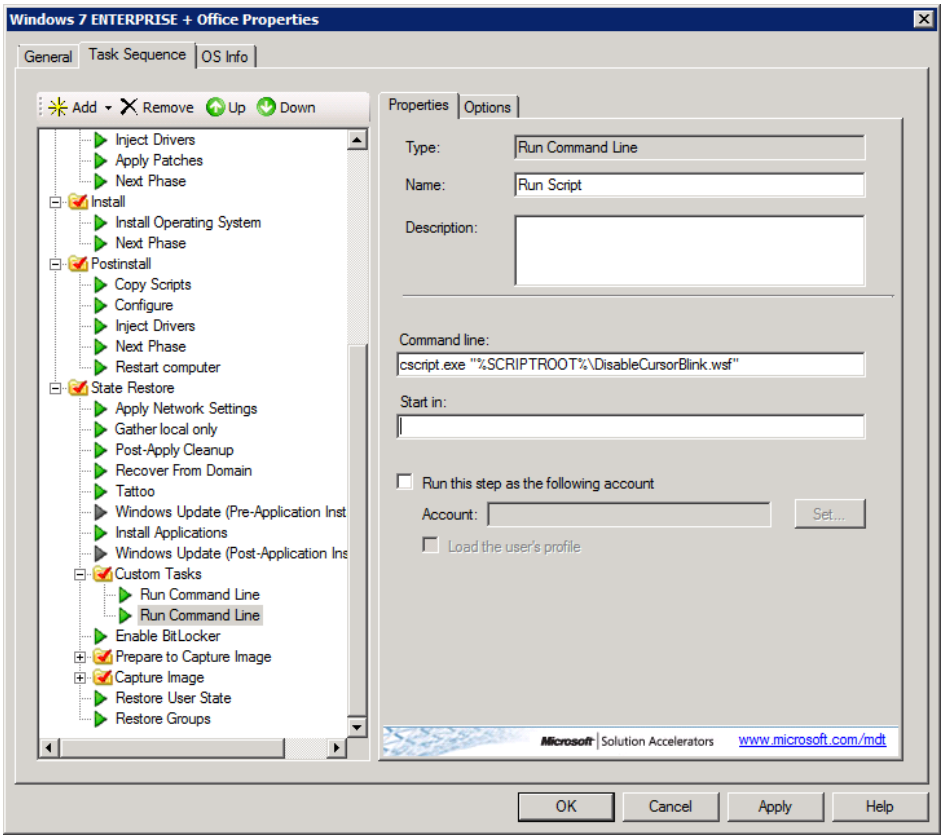

Figure 8.6: Incorporating a script into a task sequence.

More important for this discussion is where you'll store those scripts once created. For scripts to be delivered and invoked on desktops during installation, they need to be stored in the %SCRIPTROOT% folder. This folder is found on the MDT server, typically in the \Scripts subfolder of your deployment share.

A view of the Options tab can be seen in Figure 8.7. For particularly complex scripting needs, this tab provides a location to determine the success codes for a script as well as set conditions for when a script must be run. The success codes 0 and 3010 are inserted by default. A success code of 0 generally means that the script executed successfully, with a success code of 3010 generally representing a successful execution with the need for a reboot. Additional codes can be added in the field if your custom script is configured to supply them as it exits.

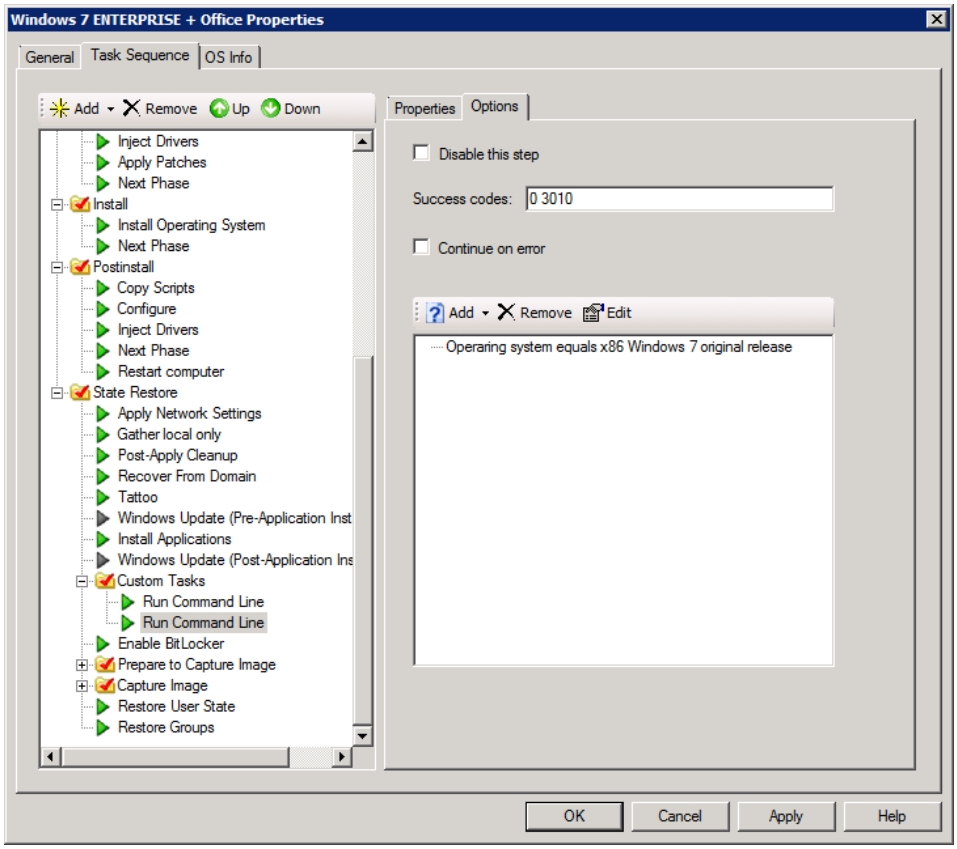

Figure 8.7: Adding success codes and conditionals.

Also seen in the lower-right of Figure 8.7 is an area where conditional statements can be added to determine whether a script should be run. Scripts are only run when conditionals in this box resolve to True.

Tuning Personal Desktops vs. Pooled Desktops

It is worth stating again that this configuration tuning process is important for Windows 7 to function best within your selected VDI platform. You'll find, however, that which configurations you'll want to tune are impacted heavily by the methods in which Windows 7 will be deployed to users. Two common methods are through the use of what are generally called *personal desktops* and *pooled desktops*.

Different vendors refer to personal and pooled desktops by different names. From a general perspective, however, the personal desktops approach refers to delivering the exact same desktop instance to each user each time. That desktop is considered the user's personal one with no other user having access to it. This approach is most similar to physical desktop deployment, where a one-to-one mapping exists between each user and their assigned desktop.

Pooled desktops are quite different. A pooled desktop is one that has been generalized and added to a pool with others that are similarly configured. As a user logs in, that user is assigned the next available desktop in the pool. As a result, they are never guaranteed to get the same desktop every time, and in fact rarely do. When well-engineered, this practice adds a significant amount of flexibility to VDI deployments, as desktops can be easily destroyed and re-created when problems occur.

To enable this flexibility, pooled desktops are typically configured using a layered approach. They also commonly make use of desktop clones, which are low-volume snapshots from a central, core image that contain only changes. Their core OS is installed with very basic settings and core configurations. Applications and user state information is then layered over the top as the desktop is provisioned to the user.

When well-engineered and well-optimized, pooled desktops tend to enjoy a much lower cost of ownership than do personal desktops. Pooled desktops tend to enjoy a higher density than personal desktops. They can be much more easily destroyed and re-created because they are based on clones from a core image. These tactics can significantly reduce the storage costs associated with VDI while increasing its user density. As a result, many of the tuning suggestions you will find tend to relate to the pooled desktop architecture. If you intend to deploy personal desktops, you might want to enable more personalization elements. Doing so will mean leaving a greater number of services and processes enabled.

Preparing and Templatizing the Deployed Virtual Machine

Desktops that will be used for VDI deployments are generally cloned (in pooled environments) or directly copied (in personal environments) from the reference virtual machine. Most VDI solutions first require converting the virtual machine into a template prior to deployment. This conversion process is a protective measure that restricts the virtual machine to be used only as a source for replication.

No matter which desktop approach you plan to use, reference virtual machines must be generalized prior to replication. Depending on your VDI solution, the generalization process will use the native SysPrep utility or a vendor-supplied utility. Notwithstanding, both SysPrep and the vendor-supplied tools achieve the same goal of removing system personality elements such as name, IP address, and SID, among others. Vendor-supplied tools will typically add configurations that prepare the virtual machine for VDI distribution.

Automating Windows 7 Installation: Start to Finish

And now we finally navigate to the end of this story. Starting with the very basics of OS installation, you've now experienced the vast majority of tools and techniques that layer on top of each other to create full automation. Yet even as this story concludes, it merely scratches the surface of the in-the-field tips and tricks that other deployment pros have learned. Regardless of whether they are specialized task sequences, scripts to add to them, or other nifty tips and tricks, all of these further automate this process while making your life easier.

Best of luck with your Windows 7 deployment project. With the foundations firmly grasped, you're well prepared to be successful in upgrading your environment to Microsoft's newest desktop OS.

Download Additional eBooks from Realtime Nexus!

Realtime Nexus—The Digital Library provides world-class expert resources that IT professionals depend on to learn about the newest technologies. If you found this eBook to be informative, we encourage you to download more of our industry-leading technology eBooks and video guides at Realtime Nexus. Please visit http://nexus.realtimepublishers.com.