Realtime publishers

Automating Windows 7 Installation for Desktop and VDI Environments

Greg Shields

Copyright Statement

Chapter 2: Creating a Universal Image that Installs Everywhere

As you complete the steps from the previous chapter, you're absolutely ready to start deploying Windows 7 images. They're not terribly customized images yet, but they'll work. Once deployed to desktops, these images will look almost exactly like the generic Windows 7 installation you would install directly from the DVD media.

There's nothing wrong with a generic image, especially considering that many of us have used the walk-around-the-office-DVD-in-hand approach for years. But there's also nothing particularly automated about them. Worse yet, these images aren't truly universal. If the computers to which you're targeting the images need a set of drivers that aren't on the DVD media, they might not work properly. Let's think for a minute about what this solution lacks:

- **Driver injection.** Windows 7 comes with a massive stack of drivers right inside its DVD media and through Windows Update. But computers today often need their own custom drivers over and above the defaults. Our solution doesn't yet have a way to inject those custom drivers into our images.
- **Automating the WinPE phase.** Remember that deploying an image requires a boot image as well as an install image. Currently, our boot image asks us a set of questions we'd prefer not to answer, things like locale, keyboard, username and password to deployment server, operating system (OS) to install, and disk partitions. Real automation means answering these questions beforehand and letting our solution do the work.
- **Automating the set up Windows phase.** Even after answering all of WinPE's questions, there's still a full second set of questions required by the Windows installation itself. The *Set Up Windows* wizard asks for a username and password, accept the license agreement, configuration settings for Windows Update, identify network settings, and set the time and time zone settings. We also want these questions answered beforehand, so we can zip past this manual step.

Thus, our solution is indeed still incomplete. It hasn't evolved yet to using a universal image, one that can install everywhere. Nor is it fully automated so that it takes care of the heavy lifting on our behalf.

There's good news: It is possible with today's Windows 7 deployment technologies to add these three capabilities with not much extra effort. I'll show you how to do just that in this chapter. As with the previous chapter, I'll be supplying you with a receipt to get started. I'll then point out a few places where you can explore on your own for even greater automation. Let's begin with the drivers.

Step Four: Dealing with Drivers

Windows Deployment Server (WDS) got a very nice new feature in Windows Server 2008 R2. That new feature takes most of the pain out of automatically injecting drivers into a deployment. Navigate over to the WDS console in Server Manager, and scroll down to its Drivers node. Here you can add sets of drivers, called *Driver Packages*, to your WDS server. Those Driver Packages contain the drivers that a computer might want. Remember Plug and Play? Windows' Plug and Play supplies the automation framework for installing these drivers inside a Windows deployment.

Plug and Play: Awesome for Windows Installations

During Windows installation, the real power of Microsoft's Plug and Play shines. You already know that Plug and Play is the service that watches for new hardware to be connected. When it detects new hardware, it matches the hardware component's characteristics to the drivers available. When it finds a match, the correct driver is automatically installed, making the hardware ready for use.

Although you're used to seeing its actions when you plug in a new device, Plug and Play is also in action during the install process. During install, Windows invokes Plug and Play to detect the installation hardware. The correct drivers are then installed if they're available. If they're not, Windows uses a generic driver when one is available.

In the end, what you need is a mechanism to make custom drivers available during the install. If they're available, Windows will take care of the rest.

Drive Packages enable you to make custom drivers available during install. Once created and added to WDS, Plug and Play will find and install the custom drivers as the OS deployment begins.

Unpacking Drivers

The hard part here is getting drivers into a format that WDS can use. WDS cannot work with drivers that are packaged into .EXE or .MSI files. Drivers need to be extracted from their .EXE or .MSI file so that WDS can find each driver's .INF file.

Remember how most drivers, when they're fully unpacked, arrive with some combination of .SYS, .CAT and .INF files? You can consider the .INF file to be a kind of instruction manual for installing the driver. It is the .INF file, usually one for each driver, that WDS needs to find. It needs the other files as well, but consider the .INF as your starting place. Everything else WDS needs is usually found in the same location.

As you can imagine, locating the .INF can be difficult. Finding it is sometimes a game of educated guessing. Sometimes drivers arrive with their .INF files already extracted. Other times, they are packaged into a .CAB or .ZIP file. Your mileage will vary, with each driver being a little different.

Unpacking the VMware Tools' Drivers

Don't worry if this unpacking process is a bit confusing the first time around. Most of us are used to using install files to automate everything after we double-click. Unfortunately, that double-click doesn't work for this specific need. Let me step you through a quick example that can illustrate one mechanism to locate the files you need.

If you're following along using VMware Workstation as your lab environment, you know that VMware's custom drivers are found in the VMware Tools. Every OS that runs inside VMware Workstation needs its VMware Tools installed if it's to work properly. For Windows, those tools are found in the file *C:\Program Files (x86)\VMware\VMware Workstation\windows.iso*.

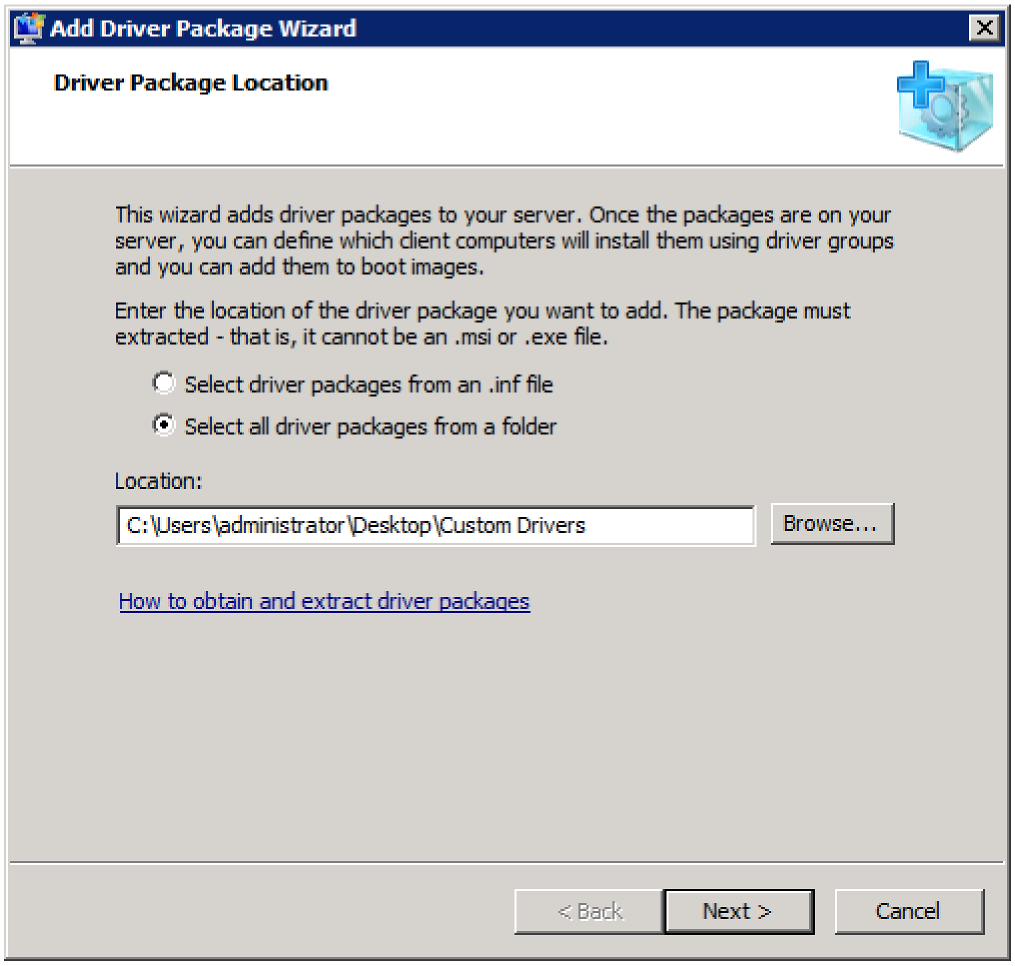
Mount that ISO using a tool such as Virtual Clone Drive, and take a look at its contents. Inside are two .MSI files that contain the VMware Tools' drivers: *VMware Tools.msi* and *VMware Tools64.msi*. You want the contents of these files. You can extract them to a folder by running an administrative install using the command:

```
msiexec /a VMware Tools64.msi
```

Doing so will launch a mini-setup that asks for a location to store the unpacked drivers. Store them somewhere they can be later accessed by WDS. You'll use these drivers in the next step.

Caution: Although this process makes for a great example, don't use this process to install the VMware Tools on production computers. VMware strongly recommends installing the tools through their usual mechanisms inside VMware Workstation.

Once you've unpacked drivers, you can add the entire folder's worth of them in one fell swoop by right-clicking *Drivers* in the WDS console, and choosing *Add Driver Package*. Figure 2.1 shows the screen that appears. You can select either a single .INF file or tell WDS to search a folder for every driver package it finds. I've chosen the second option because it's easier and adds everything at once.


Add Driver Package Wizard

Driver Package Location

This wizard adds driver packages to your server. Once the packages are on your server, you can define which client computers will install them using driver groups and you can add them to boot images.

Enter the location of the driver package you want to add. The package must extracted - that is, it cannot be an .msi or .exe file.

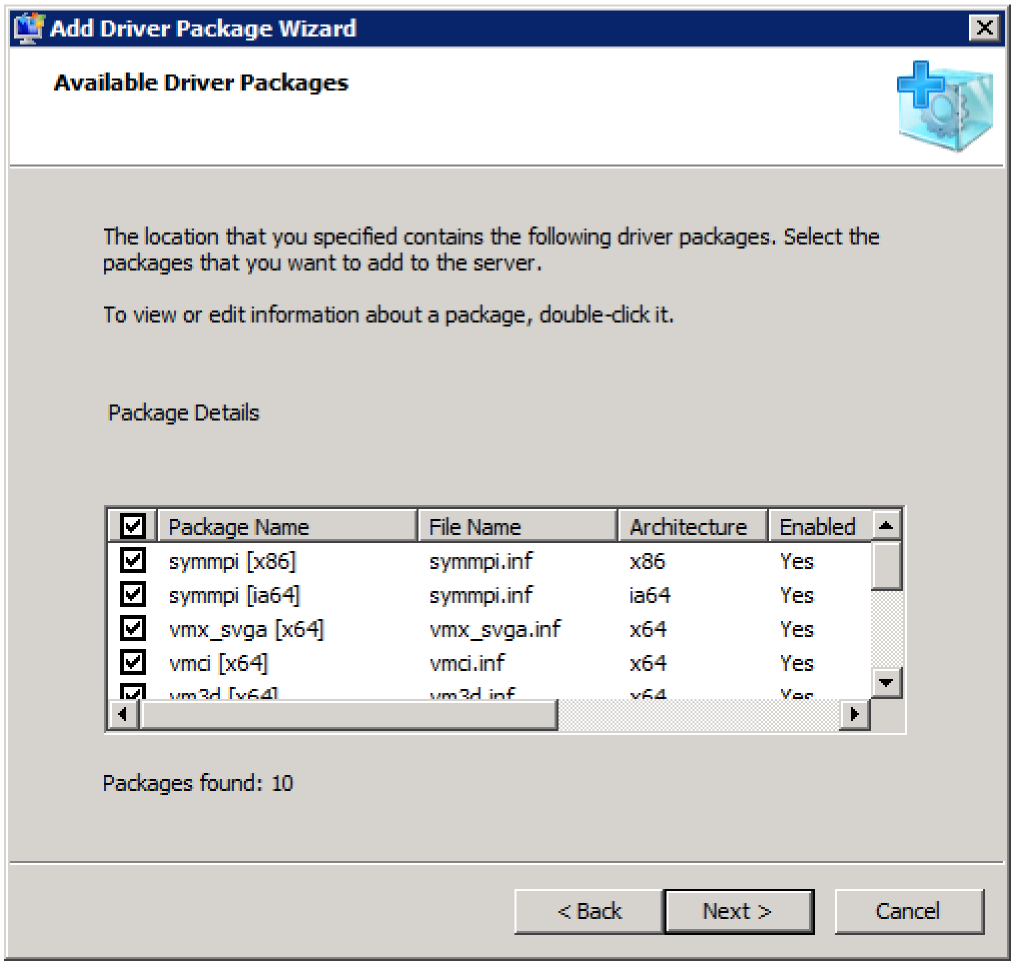
○ Select driver packages from an .inf file

● Select all driver packages from a folder

Location: C:\Users\administrator\Desktop\Custom Drivers

Figure 2.1: Driver package location.

The wizard's second page provides a place to select the drivers to add from the set WDS has found (see Figure 2.2). Some of these you won't want. For those, you have two options: clear the check box to prevent a driver's package from being loaded to WDS, or double-click the driver and change its status to *Disabled* if you want it loaded but not enabled for deployment. Double-clicking individual drivers also gives you more details about that driver, its files, and other characteristics.



Available Driver Packages

The location that you specified contains the following driver packages. Select the packages that you want to add to the server.

To view or edit information about a package, double-click it.

Package Details

✓	Package Name	File Name	Architecture	Enabled
✓	symmpi [x86]	symmpi.inf	x86	Yes
✓	symmpi [ia64]	symmpi.inf	ia64	Yes
✓	vmx_svga [x64]	vmx_svga.inf	x64	Yes
✓	vmci [x64]	vmci.inf	x64	Yes
✓	vm3d [x64]	vm3d.inf	x64	Yes

Packages found: 10

Figure 2.2: Available driver packages.

Click through the remaining wizard pages and add the drivers to the default *DriverGroup1* to complete the wizard. DriverGroup1 is the default group that can be used by all computers as an installation is deployed. Thus, right out of the box any driver that is contained in DriverGroup1 will automatically be available for any deployment to any computer.

You might at this point want to add to DriverGroup1 the set of drivers you know your desktops will need. Once added, try a test installation to see whether those drivers are successfully installed during a deployment. You'll be able to tell by logging into a desktop as Administrator after deploying an OS. Once there, check the list of drivers in Device Manager. You've done everything correctly if the right drivers appear.

Note

Although this tool is great when drivers are perfectly matched to computers, you might experience the situation where drivers conflict with each other. For example, drivers from one hardware type might be very close to those of another. As a result, the wrong set of drivers can be installed by WDS.

In this case, you'll need to configure WDS driver groups in combination with filters. WDS' filters restrict driver installations to certain hardware characteristics such as manufacturer, BIOS vendor, UUIS, OS version, and so on. They provide a way to target specific sets of drivers to specific sets of hardware.

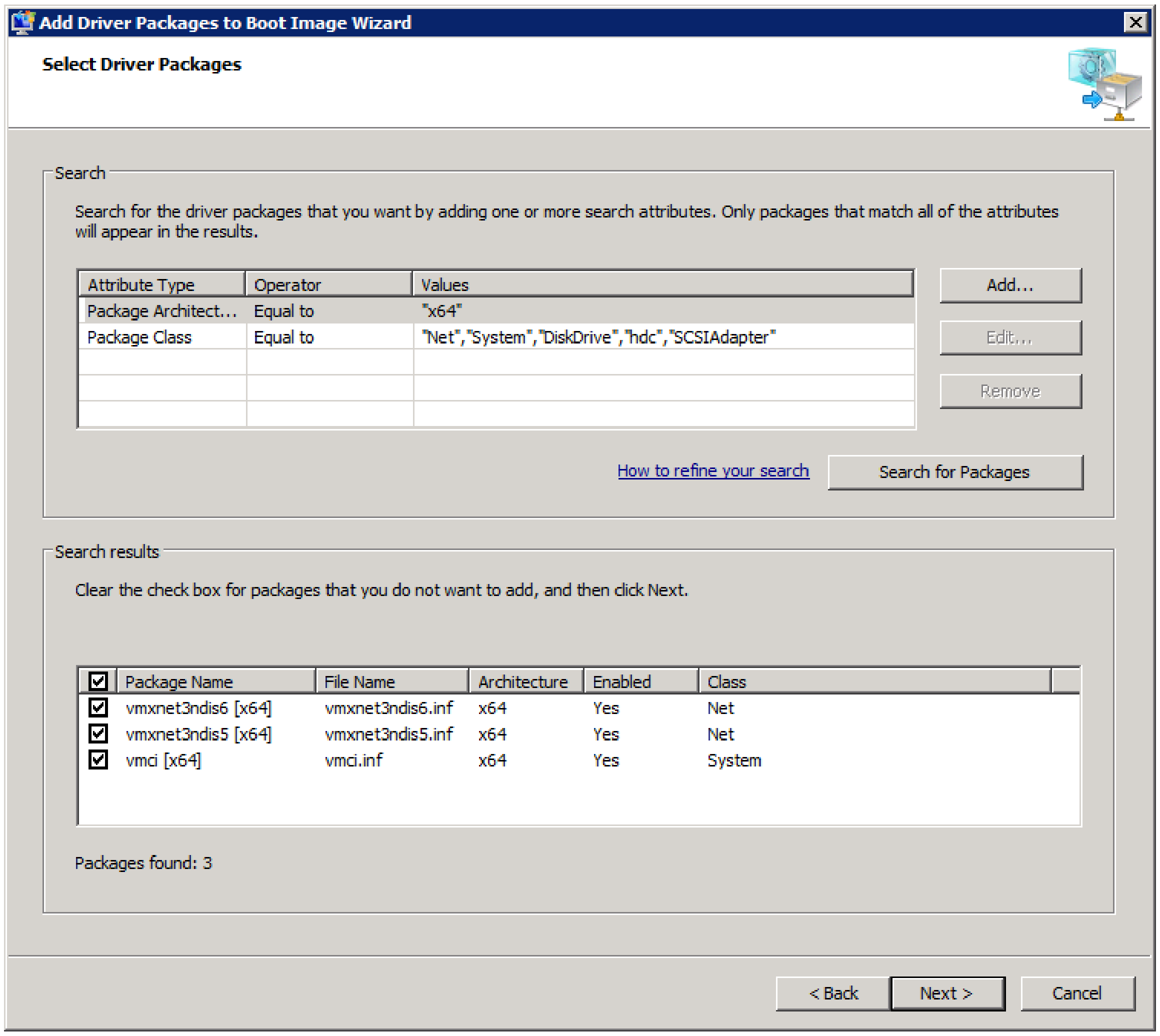
Configuring these filters is out of scope for this book; you can get more information about them at http://technet.microsoft.com/en-us/library/dd348456(WS.10).aspx.

Adding Drivers to Boot Images

Up to now, I've showed you only the steps to add drivers to install images. If your hardware will not function with the generic drivers included in WinPE, you'll need to add custom drivers to your boot image as well. This process is slightly different than with install images. The boot image process requires specifically adding drivers into the boot image rather than just making them available.

Before you start, driver packages for boot images must first be added into the *Drivers* node using the method I've already explained. Once added, inject them into a boot image by right-clicking the boot image, and choosing *Add Driver Packages to Image*. This selection brings up a multi-screen wizard for selecting the appropriate driver packages. Figure 2.3 shows the most important page in that wizard.

This page locates driver packages based on the search terms you set in the Search box. The settings in Figure 2.3 are the defaults for my standard x64 boot image. Adjust your search terms, then click *Search for Packages* to reveal the results at the bottom. Select the check boxes for the appropriate drivers, and complete the wizard to add them to the boot image.



Search for the driver packages that you want by adding one or more search attributes. Only packages that match all of the attributes will appear in the results.

Attribute Type	Operator	Values
Package Architect...	Equal to	"x64"
Package Class	Equal to	"Net","System","DiskDrive","hdc","SCSIAdapter"

Clear the check box for packages that you do not want to add, and then click Next.

☑	Package Name	File Name	Architecture	Enabled	Class
☑	vmxnet3ndis6 [x64]	vmxnet3ndis6.inf	x64	Yes	Net
☑	vmxnet3ndis5 [x64]	vmxnet3ndis5.inf	x64	Yes	Net
☑	vmci [x64]	vmci.inf	x64	Yes	System

Packages found: 3

Figure 2.3: Add Driver Packages to Boot Image Wizard.

Note

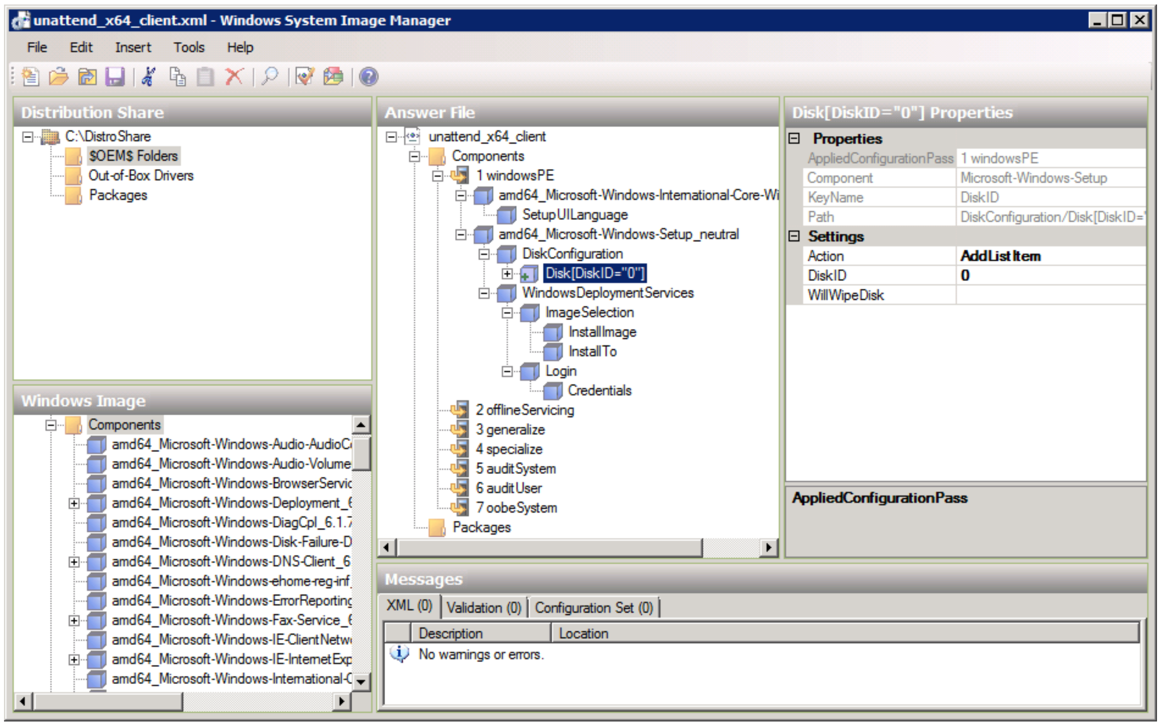
Although adding driver packages to boot images is easy, removing them is less so. Thus, it is recommended that you make a backup copy of the image before making any changes. Otherwise, a wrongly-placed driver can corrupt the boot image.

Step Five: Automating the WinPE Boot Image

With the right set of custom drivers, your images can now be made universal. That makes you ready for the next step in constructing your deployment solution: automating responses to the set of questions asked by WinPE prior to starting an OS deployment. These questions relate to locale, keyboard, the username and password WinPE needs to download an install image, OS and edition, and disk partitions.

Things get a little more challenging in this step. First, you'll need to learn two more of the acronyms in the Microsoft deployment alphabet soup: WAIK and WSIM. The WAIK is the *Windows Automated Installation toolKit*. Among many other deployment-related tools, the WAIK is the home of the *Windows System Image Manager* (WSIM). Your first job is to locate the Windows 7 version of the WAIK (they are versioned by OS). For this book, download and install the WAIK to your WDS server.

Once installed, launch WSIM. WSIM provides a workbench for creating unattended installation files. Figure 2.4 shows what WSIM looks like after completing some of the preparations I'll explain next. As you'll immediately note, it's a pretty busy interface. I'll walk through each of its panes in the next section.


unattend_x64_client.xml - Windows System Image Manager
File
Edit
Insert
Tools
Help
Distribution Share
C:\DistroShare
OEM Folders
Out-of-Box Drivers
Packages
Windows Image
Components
amd64_Microsoft-Windows-Audio-AudioC
amd64_Microsoft-Windows-Audio-Volume
amd64_Microsoft-Windows-BrowserServic
amd64_Microsoft-Windows-Deployment_6
amd64_Microsoft-Windows-DiagCpl_6.1.7
amd64_Microsoft-Windows-Disk-Failure-D
amd64_Microsoft-Windows-DNS-Client_6
amd64_Microsoft-Windows-ehome-reg-inf
amd64_Microsoft-Windows-ErrorReporting
amd64_Microsoft-Windows-Fax-Service_6
amd64_Microsoft-Windows-IE-ClientNetw
amd64_Microsoft-Windows-IE-InternetExp
amd64_Microsoft-Windows-International-C
Answer File
unattend_x64_client
Components
1 windowsPE
amd64_Microsoft-Windows-International-Core-Wi
SetupUILanguage
amd64_Microsoft-Windows-Setup_neutral
DiskConfiguration
Disk[DiskID="0"]
WindowsDeploymentServices
ImageSelection
InstallImage
InstallTo
Login
Credentials
2 offlineServicing
3 generalize
4 specialize
5 auditSystem
6 auditUser
7 oobeSystem
Packages
Disk[DiskID="0"] Properties
AppliedConfigurationPass
Messages
XML (0)
Validation (0)
Configuration Set (0)

Properties	
AppliedConfigurationPass	1 windowsPE
Component	Microsoft-Windows-Setup
KeyName	DiskID
Path	DiskConfiguration/Disk[DiskID=
Settings	
Action	**AddListItem**
DiskID	**0**
WillWipeDisk	

Description	Location
No warnings or errors.	

Figure 2.4: An example WSIM interface.

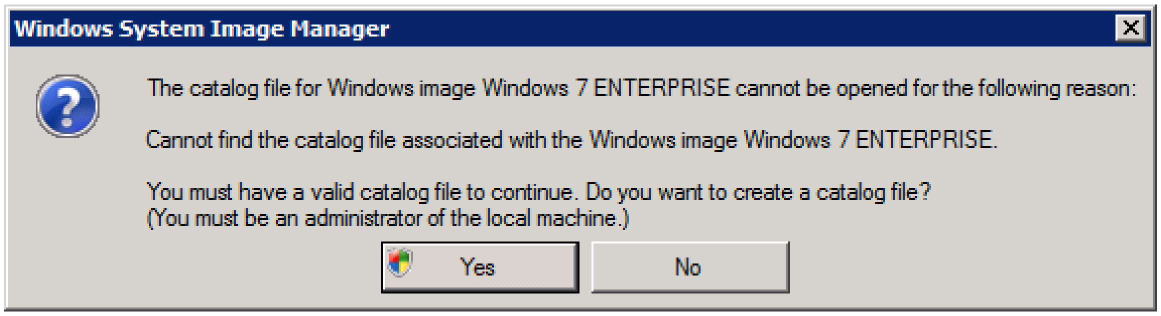
First a little background on what you're about to create. The files WSIM generates provide the "answers" to the "questions" you'll want to eliminate from your automated installation. You'll in fact use WSIM twice to do this. For its first run-through, you'll be answering the questions WinPE asks during its portion of the installation. For the second—which I'll explain in Step Six—you'll follow up with an additional set of answers that will be used by the Windows installation itself.

Preparing WSIM

To get started with WSIM, create a Distribution Share. For our purposes, create a folder called *C:\DistroShare* on your WDS server. Then, in WSIM, right-click *Select a Distribution Share* | *Create Distribution Share*. Navigate to C:\DistroShare, and click *Open* to set this as your share.

Your next action will be to export an install image from WDS to the share. This image is needed so that WSIM can analyze it to discover the questions that might need answering. Right-click an install image that you plan to eventually deploy (yes, that's an *install image* and not a boot image), and choose Export Image. This will be the image that you later plan to deploy to your desktops. I'll be choosing the Windows 7 ENTERPRISE edition image for mine. Export that image to C:\DistroShare and give it a name. Then, back in WSIM, right-click *Select a Windows image or catalog file* | *Select Windows Image*, then choose the .WIM image you just exported.

You'll be immediately prompted with the error message you see in Figure 2.5. This is OK because you haven't yet created a catalog file that is associated with this image. Create that catalog by clicking Yes. This catalog file takes a few minutes to generate as WSIM analyzes the image for all the possible questions that could be asked. Once it is complete, you'll be able to pick and choose from the questions it discovers and answer those of interest.


Windows System Image Manager

The catalog file for Windows image Windows 7 ENTERPRISE cannot be opened for the following reason:

Cannot find the catalog file associated with the Windows image Windows 7 ENTERPRISE.

You must have a valid catalog file to continue. Do you want to create a catalog file?
(You must be an administrator of the local machine.)

Figure 2.5: Cannot find a valid catalog file.

Figure 2.6 shows what the Windows Image pane will resemble once this process has completed. You'll see that a very large number of questions are available to be answered, many of which are difficult to understand and all of which have cryptic names. Many questions have sub-categories of further questions.

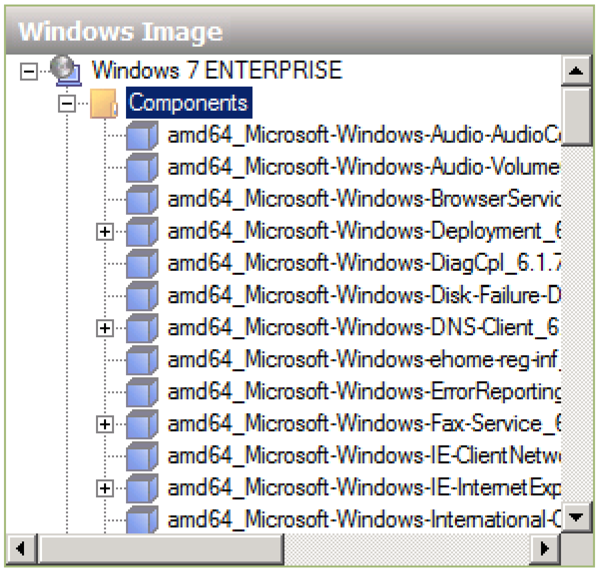

Windows Image
Windows 7 ENTERPRISE
Components
amd64_Microsoft-Windows-Audio-AudioC
amd64_Microsoft-Windows-Audio-Volume
amd64_Microsoft-Windows-BrowserServic
amd64_Microsoft-Windows-Deployment_6
amd64_Microsoft-Windows-DiagCpl_6.1.7
amd64_Microsoft-Windows-Disk-Failure-D
amd64_Microsoft-Windows-DNS-Client_6
amd64_Microsoft-Windows-ehome-reg-inf
amd64_Microsoft-Windows-ErrorReporting
amd64_Microsoft-Windows-Fax-Service_6
amd64_Microsoft-Windows-IE-ClientNetw
amd64_Microsoft-Windows-IE-InternetExp
amd64_Microsoft-Windows-International-C

Figure 2.6: WSIM's Windows Image pane.

You might spend quite a bit of time here simply finding the questions of interest. Many of these are intended only for the activities in Step Six. Others are intended for what we're doing here with WinPE in Step Five. To be honest, this process is mostly a guess-and-test activity. I'll help get you started with the minimum you'll need to fully automate this step. Later, you can add more answers as you see fit and desire more customization.

Before you can begin working with questions, you'll need to create an answer file. You'll eventually add this .XML file into WDS. Create it by right-clicking *Create or open an answer file* | *New Answer File* in the Answer File pane (see Figure 2.7).

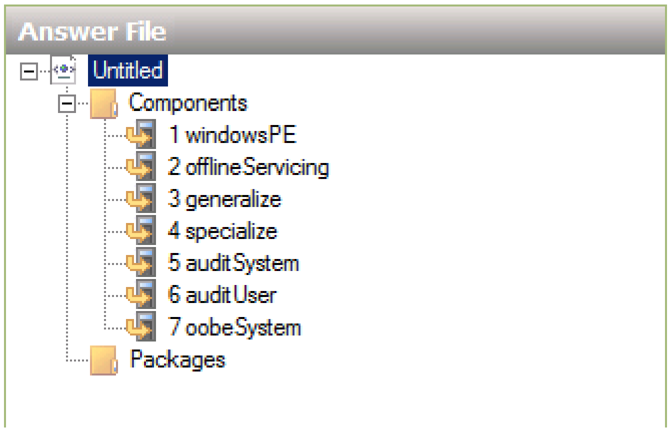


Figure 2.7: An answer file.

Note

Every answer file has seven numbered sections. These seven sections correspond to the seven "passes" of the Windows installation, starting with windowsPE and finishing with oobeSystem (OOBE stands for Out-Of-Box Experience). For Step Five in our process, we'll be working only with questions that relate to Pass 1.

Locating the Right Questions and Inserting Answers

At this point, you'll need to locate the questions you want and insert whatever answers make sense for your Windows 7 deployment. I'll kick off this process by explaining how to answer one question in detail. Then, in the next section I'll follow up with a table of the remaining questions and their answers. Remember that I am using the x64 installation of Windows 7. Questions for x64 installations begin in WSIM with *amd64*. If you will be installing the x86 version, look for those that start with *x86*.

The first question you want to answer relates to the regional language to be used by the installation. In the Windows Image pane, navigate to *amd64_Microsoft-Windows-International-Core-WinPE_{version}_neutral* where {version} is whatever version number you see. Right-click this element, and select *Add Setting to Pass 1 windowsPE* (see Figure 2.8).

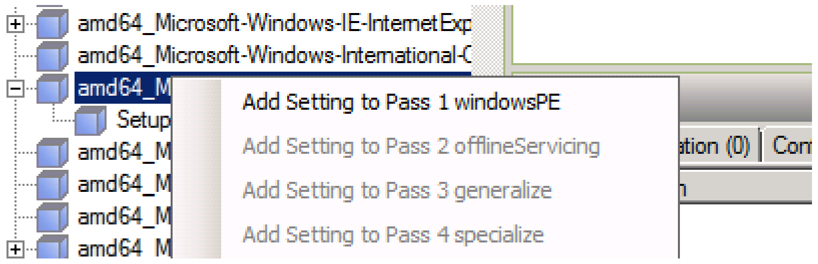
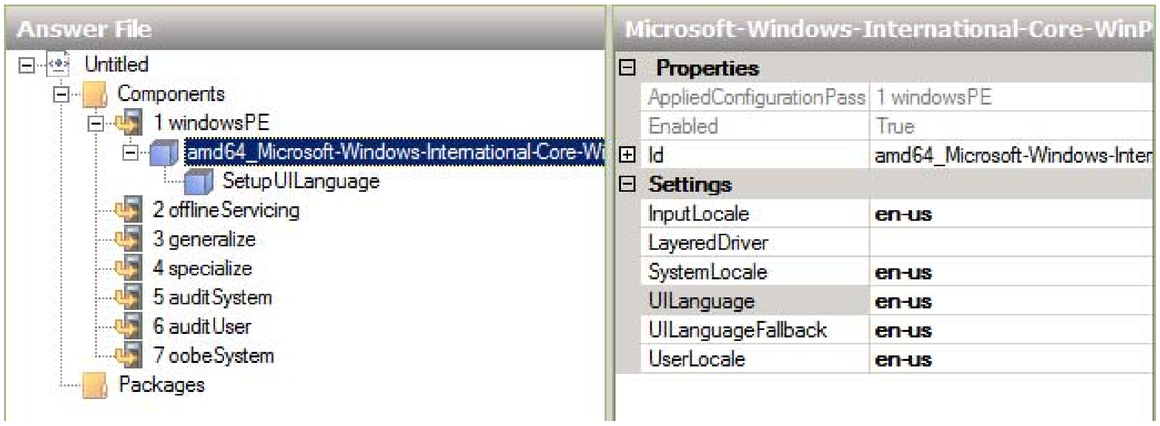

amd64_Microsoft-Windows-IE-InternetExp
amd64_Microsoft-Windows-International-C
amd64_M
Setup
Add Setting to Pass 1 windowsPE
Add Setting to Pass 2 offlineServicing
Add Setting to Pass 3 generalize
Add Setting to Pass 4 specialize

Figure 2.8: Adding a setting to Pass 1 windowsPE.

Doing so adds the question to your answer file. Once added, you'll then answer it in WSIM's upper-right pane. That pane's title will always be the name of the question you've currently selected in the Answer File pane. Figure 2.9 shows both panes with the question highlighted that I've just selected.



Properties	
AppliedConfigurationPass	1 windowsPE
Enabled	True
Id	amd64_Microsoft-Windows-Inter
Settings	
InputLocale	en-us
LayeredDriver	
SystemLocale	en-us
UILanguage	en-us
UILanguageFallback	en-us
UserLocale	en-us

Figure 2.9: Answering a question.

If your language choice is United States English, the correct answer for this question is *en-us*. Enter this value into each of the boxes except for LayeredDriver under settings in the upper-right pane.

Note

You wouldn't necessarily know this by looking, but LayeredDriver is only used by Japanese and Korean keyboards. I bring up this point to highlight how much research, guessing, and testing you'll be doing when you attempt to customize. So, yes, what you're thinking is correct: *This isn't easy.* Keep following along, and I'll at least get you started.

Also notice in Figure 2.9 that *amd64_Microsoft-Windows-International-Core-WinPE_{version}_neutral* has another question below it in the subfolder marked *SetupUILanguage*. You'll want to answer that question as well. Click to view SetupUILanguage, and enter *en-us* again next to *UILanguage* in the upper-right pane.

The Remaining Questions and Answers

You have now completed answering the first question. Table 2.1 provides all the questions and answers. For each, I'll explain the question as well as possible corresponding answers. Remember that these answers are for my environment as I've been building it in this book. Thus, yours might be slightly different. Locate and answer each of these before moving on.

Windows Image Pane (Question)	**Upper-Right Pane (Answer)**	**Explanation**

amd64_Microsoft-Windows-International-Core-WinPE_{version}_neutral	InputLocale = en-us
	SystemLocale = en-us
	UILanguage = en-us
	UILanguageFallback = en-us
	UserLocale = en-us
		This item configures the WinPE language to US English.

amd64_Microsoft-Windows-International-Core-WinPE_{version}_neutral\SetupUILanguage	UILanguage = en-us	
amd64_Microsoft-Windows-Setup_{version}_neutral\WindowsDeploymentServices\Login\Credentials	Domain Username Password	Enter the domain, username, and password of the user who connects to your WDS share (the same user as in Chapter 1, Figure 1.10).
amd64_Microsoft-Windows-Setup_{version}_neutral\Disk Configuration\Disk	DiskID = 0	This item begins working with the first disk in the computer.

amd64_Microsoft-Windows-Setup_{version}_neutral\Disk Configuration\Disk\Create Partitions\CreatePartition	Extend = true
	Order = 1
	Type = Primary
		This item creates a single primary disk to install Windows.

amd64_Microsoft-Windows-Setup_{version}_neutral\Disk Configuration\Disk\Modify Partitions\ModifyPartition	Active = true
	Format = NTFS
	Label = Windows
	Letter = C
	Order = 1
	PartitionID = 1
		This item modifies that partition to create the C: drive as the first NTFS drive and partition.

amd64_Microsoft-Windows-Setup_{version}_neutral\WindowsDeploymentServices\ImageSelection\InstallTo	DiskID = 0
	
	PartitionID = 1
		This item installs Windows to the disk and volume created in the rows above.

amd64_Microsoft-Windows-Setup_{version}_neutral\WindowsDeploymentServices\ImageSelection\InstallImage	Filename ImageGroup ImageName	See the note below.

Table 2.1: Step Five's remaining questions and answers.

Note

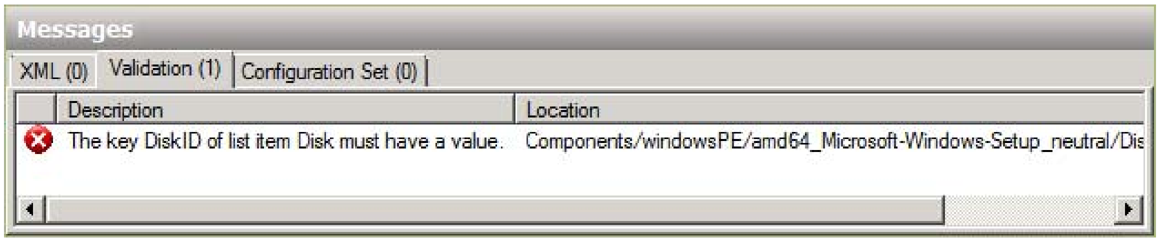
The InstallImage question is a special case. If you want complete automation, enter the filename, Image Group, and Image Name of the image you want to deploy from WDS. You can get this information by reviewing the properties of the image in WDS. In my case, the filename will be *install.wim*, the ImageGroup will be *Windows 7 Default*, and the ImageName will be *Windows 7 ENTERPRISE*.

If you do not enter values, you will be given the option to select an image at the time of installation. Thus, there's a tradeoff between full automation and being able to choose an image at every OS deployment. Remove the question entirely from the answer file if you won't be supplying answers.

Validating, Saving, and Adding the Answer File to WDS

The hard part is done! Three easy steps are next: validating the answer file, saving it, and finally entering it into WDS. The first step is validation. Run a validation by selecting *Tools | Validate Answer File*.

Your answer file should validate with no warnings or errors. If one exists, WSIM will tell you what needs to be fixed. For example, Figure 2.10 shows the error message you would see if your DiskID item under Disk was missing a value. Fix any errors and re-run the validation again until it reports *No warnings or errors*.

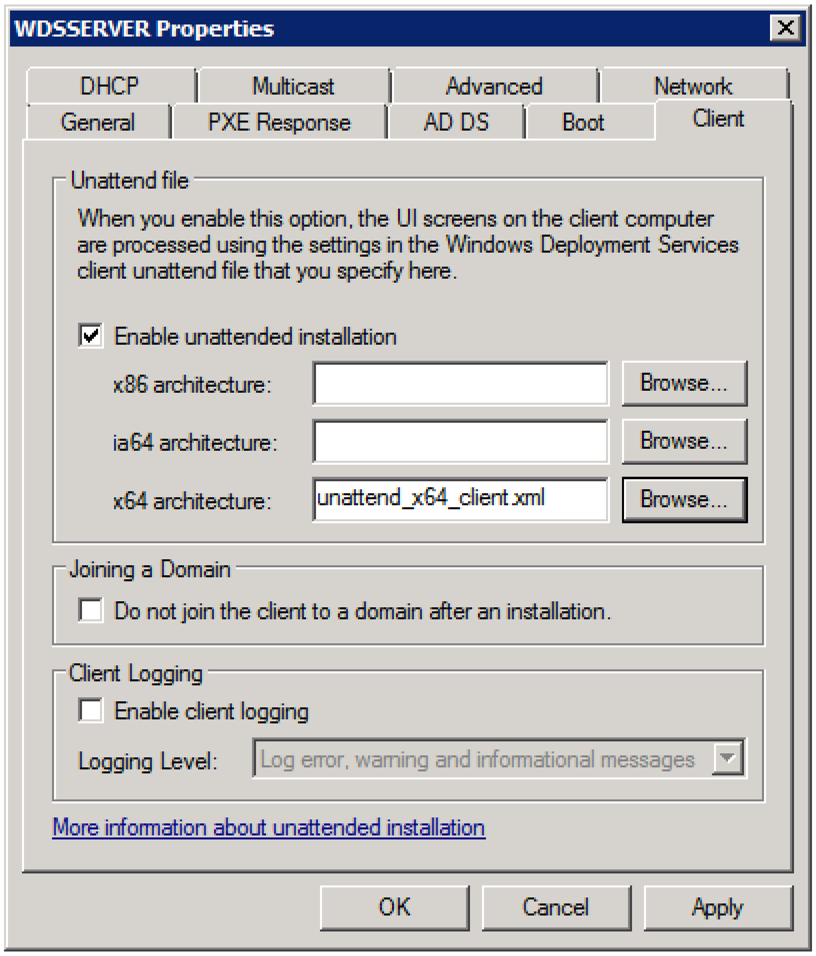


	Description	Location
✖	The key DiskID of list item Disk must have a value.	Components/windowsPE/amd64_Microsoft-Windows-Setup_neutral/Dis

Figure 2.10: A validation error.

Save the answer file to the C:\RemoteInstall folder on the WDS server by selecting *File | Save Answer File*. Because this answer file will be located under the Client tab in WDS, let's name this file *unattend_x64_client.xml*.

Answer files for WinPE are configured on a per WDS server basis and not a per boot image basis. Thus, a single answer file will be used for all deployments of a particular architecture (x86, ia64, or x64) for the entire WDS server. You can attach answer files you just created by navigating back to the WDS console, and right-clicking the *server name* | *Properties*. Under the client tab (see Figure 2.11), select the *Enable unattended installation* check box, and click *Browse* next to the correct architecture (x64 in my case). Provide the path to the unattend_x64_client.xml file, and click OK.



Unattend file

When you enable this option, the UI screens on the client computer are processed using the settings in the Windows Deployment Services client unattend file that you specify here.

[✓] Enable unattended installation

x86 architecture:

ia64 architecture:

x64 architecture: unattend_x64_client.xml

Joining a Domain

[ ] Do not join the client to a domain after an installation.

Client Logging

[ ] Enable client logging

Logging Level: Log error, warning and informational messages

Figure 2.11: Adding the unattended installation file to WDS.

At this point, you might want to re-run another OS deployment, just to see how the installation has now automated its first steps. As you'll discover, you won't be asked any questions until you get to the Set Up Windows wizard after the installation completes. Fixing that is a job for Step Six.

Step Six: Automating the Set Up Windows Phase

Getting rid of that pesky Set Up Windows wizard requires one more step in fully automating the installation. By combining this step, what we accomplished in Step Five, and the driver work in Step Four, you've fulfilled this chapter's goal of creating a universal Windows image that installs anywhere.

You already know that WSIM provides you an almost-too-much-to-handle list of possible questions and answers that customize your Windows installation. At this point, however, what you want is that universal image that installs everywhere with no added effort. This step will show you the few additional questions and answers WSIM can manage in order to eliminate the Set Up Windows wizard. Any further customization I'll leave to your own curiosity. You can spend literally *days* tweaking unattend files in WSIM to get the process just right.

Getting through Set Up Windows means addressing four areas: Setting the time zone, accepting the EULA, setting up the network location, and creating a local administrator and password. Each of these steps can be pre-answered through the same WSIM tool you've already experienced in Step Five. Different here is where you'll attach that file once created. Rather than attaching it to the *Clients* tab for the WDS server, you get to apply your unattend file directly to the image you plan to deploy.

Rather than go through the process again for using WSIM, let me give you a second table of questions and answers that you'll need. As in Step Five, use WSIM to create an unattend file with these questions and whatever answers make sense for your environment. However, there is one important difference. For Step Six, you'll be adding your questions and answers to different passes in the installation process, specifically Pass 4 and Pass 7, rather than Pass 1. Table 2.2 includes the correct passes to get you going. Remember that, as before, these answers are for my environment as I've been building it. Yours might be slightly different.

Windows Image Pane (Question)	**Upper-Right Pane (Answer)**	**Explanation**
amd64_Microsoft-Windows-Shell-Setup_{version}_neutral (Pass 4)	ComputerName = %MACHINENAME% TimeZone	Setting ComputerName to %MACHINENAME% will pass through the name you set in WDSs *Name and Approve*. Set TimeZone to your correct time zone, such as *Mountain Standard Time*.[1]

amd64_Microsoft-Windows-International-Core_{version}_neutral (Pass 7)	InputLocale = en-us
	SystemLocale = en-us
	UILanguage = en-us
	UserLocale = en-us
		This item configures the Windows language to US English.

[1] See technet.microsoft.com/en-us/library/cc749073(WS.10).aspx for a list of applicable time zone strings.

Windows Image Pane (Question)	Upper-Right Pane (Answer)	Explanation

amd64_Microsoft-Windows-Shell-Setup_{version}_neutral\ oobe (Pass 4)	HideEULAPage = true
	HideWirelessSetupIn OOBE = true
	NetworkLocation = work
	ProtectYourPC = 1
		Hides the EULA and wireless setup screens, sets the network location to work, and enables Automatic Updates.

amd64_Microsoft-Windows-Shell-Setup_{version}_neutral\ UserAccounts\LocalAccounts\ LocalAccount (Pass 7)	DisplayName = LocalAdmin
	Group = Administrators
	Name = LocalAdmin
		This item adds a local administrator account named *LocalAdmin*.

amd64_Microsoft-Windows-Shell-Setup_{version}_neutral\ UserAccounts\LocalAccounts\ LocalAccount\Password (Pass 7)	Value = {Password}	This item configures the password for the administrator account created earlier.

Table 2.2: WSIM questions and answers for Step Six.

As before, validate this file and save it to complete this step. I'll name my file *C:\RemoteInstall\unattend_w7_enterprise.xml* as this file corresponds specifically to the Windows 7 ENTERPRISE deployment in WDS.

Note

I mentioned that you could add customization. One place to do so is in the subfolders of *amd64_Microsoft-Windows-Shell-Setup_{version}_neutral*. You can either further personalize your installation by answering their questions, or remove them from the file by right-clicking each, and choosing Delete.

Be aware that your validation may display a set of yellow warnings if these subfolder items are not populated with information or not removed from the unattend file. So plan on either answering them or deleting them so that validation completes successfully.

As in Step Five, the final part of Step Six is to attach this file to its correct location inside WDS. To do so, navigate to the WDS console in Server Manager, then go to *Install Images* and find the install image for the unattend file you just created. Right-click that install image, and view its properties.

You'll see a check box at the bottom of the properties window (see Figure 2.12). Select this check box, and click *Select File*. Provide the path to the unattend_w7_enterprise.xml file, and click OK.

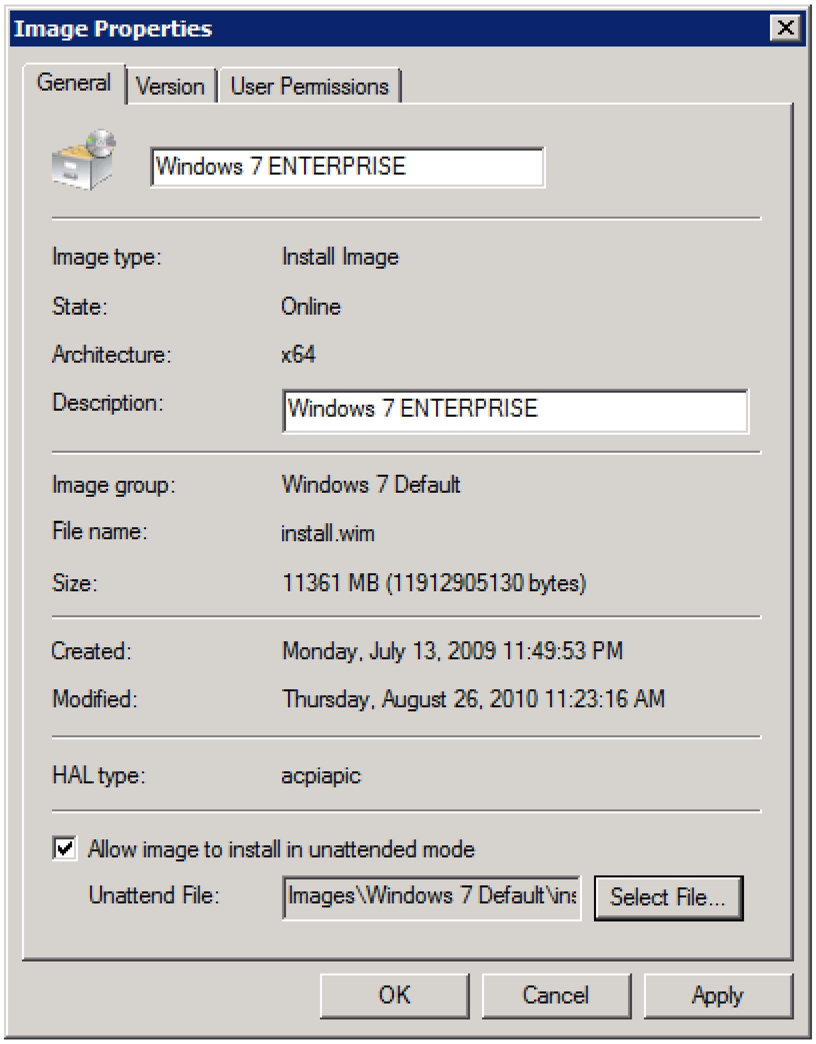


Image Properties

General | Version | User Permissions

Windows 7 ENTERPRISE

Image type: Install Image
State: Online
Architecture: x64
Description: Windows 7 ENTERPRISE

Image group: Windows 7 Default
File name: install.wim
Size: 11361 MB (11912905130 bytes)

Created: Monday, July 13, 2009 11:49:53 PM
Modified: Thursday, August 26, 2010 11:23:16 AM

HAL type: acpiapic

[✓] Allow image to install in unattended mode
Unattend File: Images\Windows 7 Default\ins

Figure 2.12: An image's properties.

You're done! At this point, re-run your installation one more time and amaze yourself with a Windows 7 OS that fully deploys without additional input.

Time for Applications

In two chapters, we've started with little more than the manual installation of Windows and added six steps of automation. You now know how to extend the basic image to virtually any kind of hardware through drivers and Plug and Play. You also know how to automate the WinPE and Set Up Windows portions of the installation so that you can kick off a deployment and come back to a CONTROL+ALT+DELETE screen.

Yet there are still a few things that remain ominously missing at this point. The first and most obvious of these are those pesky applications. A Windows computer isn't terribly useful for users if they've got no applications to run on that computer. That will be the topic for the next chapter. In it, I'll show you different approaches for getting applications onto computers, some of which might surprise you.