Chapter 7. Image Deployment

In the previous chapter, we discussed driver management in MDT. We discussed how to perform offline servicing in order to inject drivers into our image post-capture. We also discussed how plug and play works with MDT and driver profiles using variables in MDT to force drivers into an installation based on properties such as model number.

In this chapter, we will go over how to deploy our image, both for capturing a reference image, as well as doing deployments in test and production environments. Various caveats and situations will be covered and several real-world scenarios will be examined as well.

We will cover the following topics in this chapter:

- Reference image deployment and image types
- Virtual machine creation
- Deployment steps
- Deployment share
- Deployment scenarios and network considerations

Reference image deployment

In previous chapters, we've discussed drivers, `CustomSettings.ini` configuration, task sequence, and many other items. All these concepts are utilized here as we build our reference image deployment task sequence and virtual machine. In our examples in this chapter, we're going to be utilizing Hyper-V as a virtualization host for simplicity and cost; other solutions can be used, but they add a complication layer in terms of drivers that need to be inserted into both the **Windows Preinstallation Environment** (**Win PE**) and actual driver store for the image itself. However, Hyper-V can pose the same concern with driver-versioning needs.

TIP

What this means is the native Hyper-V drivers that are shipped with Windows 7, for instance, will not work in a Hyper-V virtual machine hosted in Windows Server 2012 R2, for example. Back-level drivers will still need to be provided so that the Win PE and installation task sequence can see the network, and thus communicate with the deployment share.

When we design our reference image, it behooves us to review the business needs and consider the scenarios of thick, thin, and hybrid images. There are many scenarios and business cases where Windows is used, and a blanket statement that one image type is better than another is a slippery slope. Let's review the options and discuss when it would be proper to implement the specific type of image.

Thick image

The thick image is one that contains all the applications needed by the overall business. Depending on the bandwidth constraints connecting the sites for image replication, this may be a viable and correct option.

For example, I had a customer implement the thick image due to limited connectivity between their deployment sites. Replicating more than a single image took a very long time to finish. Therefore, the solution was to have a master thick image. They then had multiple task sequences that used the WIM and uninstalled select applications based on the appropriate scenario (laptop, desktop, and so on).

Thin image

The thin image contains a patched WIM of Windows and scant else. It is a barebones kit, where applications are installed (or streamed via an application provisioning/virtualization layer) at deployment, or even at the initial run of the application. These deployments tend towards the campus deployment scenario, where high bandwidth and low latency are present. Some VDI implementations go this route, particularly with the application streaming components in place.

Another use case for this would be flexibility. You only need to rebuild a single image if you need to apply additional updates, and so on. This makes maintenance of your deployment images much easier and still gives the flexibility of multiple task sequences for different business needs.

Hybrid image

Perhaps the most common of deployment methodologies, the base thin image is crafted with a task sequence that (in addition to patching) adds universal applications that any user at the business is licensed for and might feasibly use. An example would be a build with Windows Updates, Office 2013, and Updates applied, but not the HR application nor the accounting one. This type of image usually makes the most sense from licensing, bandwidth, servicing, and deployment perspective.

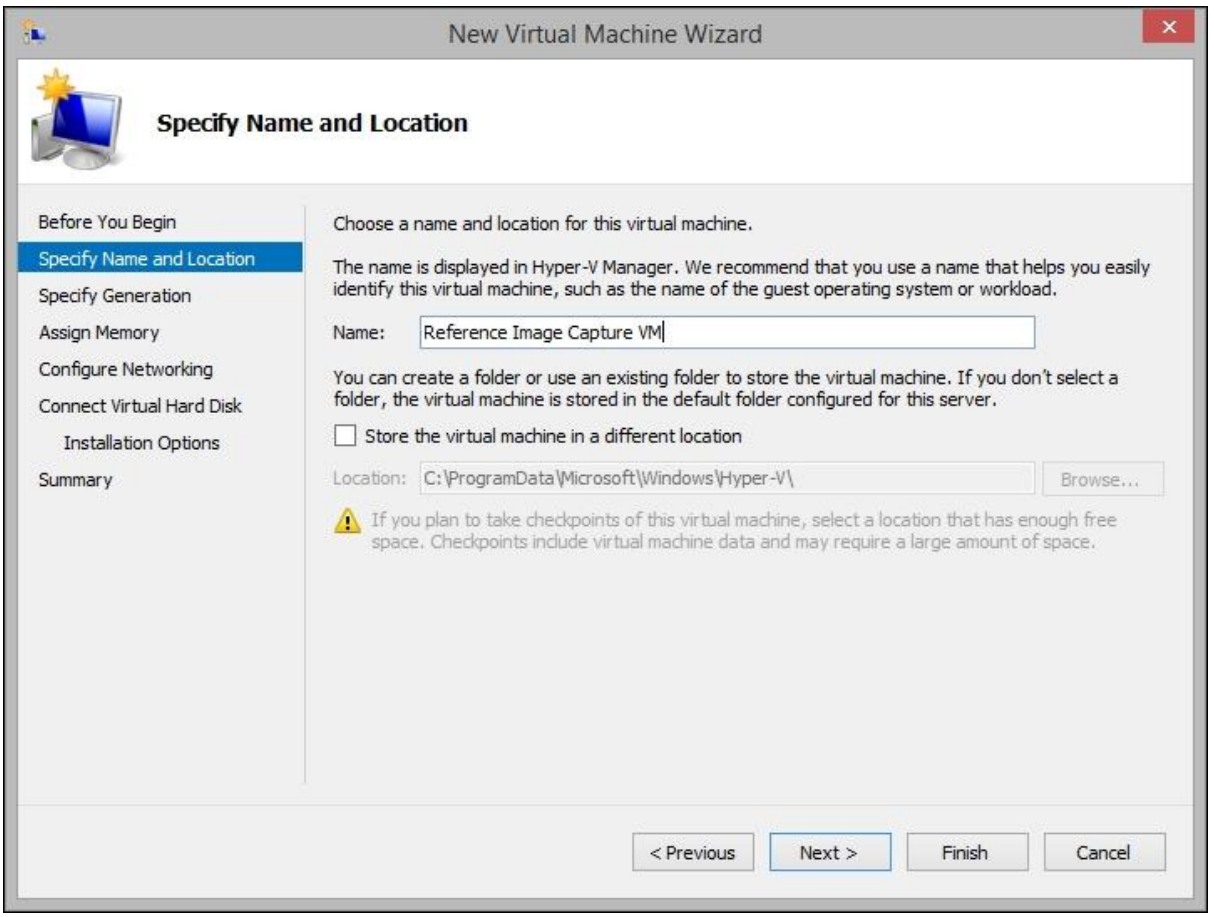
Now the reader should keep in mind that any of these images can be the subject of many task sequences that change `CustomSettings.ini`, `Unattend.xml`, and so

on, and customize the deployment in whatever manner needed. Therefore, in considering the type of image to craft, consider this a skeletal design onto which you will later craft many different and diverse task sequences against (potentially, some shops get by with one or two images).

Virtual machine creation

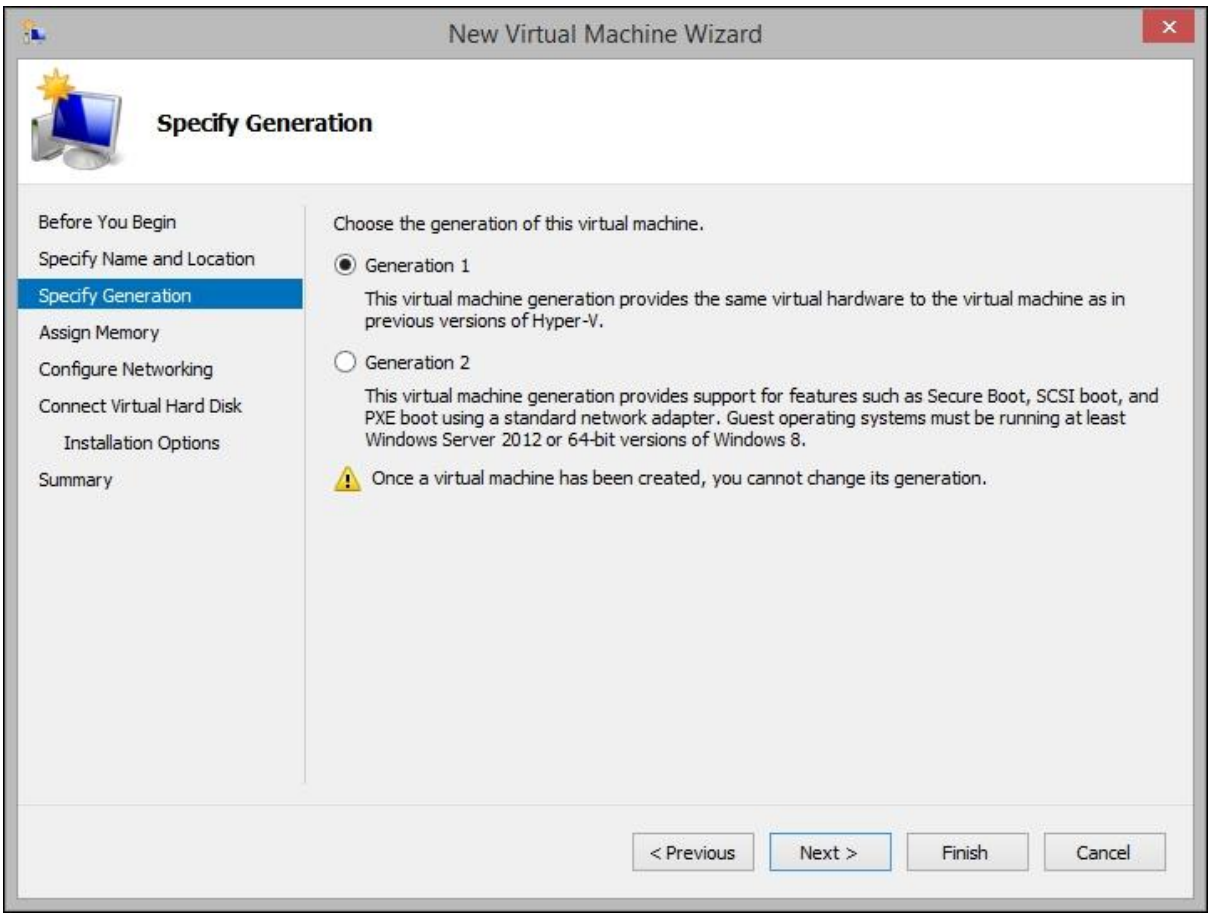
The screenshots and text are taken from Windows 8.1 Update 1, which is quite similar to Windows Server 2012 R2 as well. Most concepts will apply to down-level Hyper-V hosts as well though.

1. The first action item is to create a virtual machine. Generally speaking, a virtual machine with one core and 2 GB of RAM is adequate for our purpose. We'll walk through the wizard and create our virtual machine as follows:



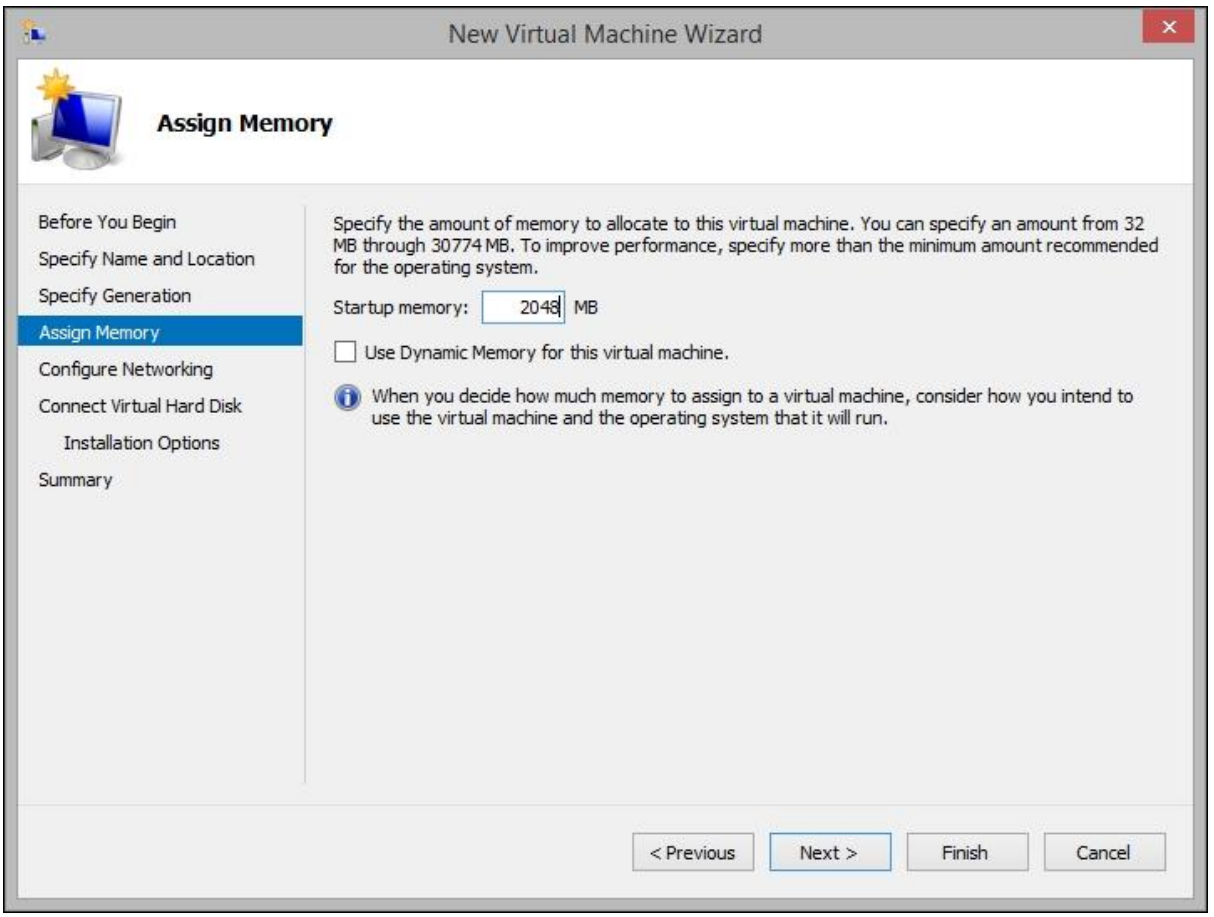
Virtual Machine name

2. In the next step, we'll want to select **Generation 1**. This is to support both x86 and x64 versions of Windows, and also Windows 7 and Server 2008 R2, as well as newer OSes:



Generation 1 selection

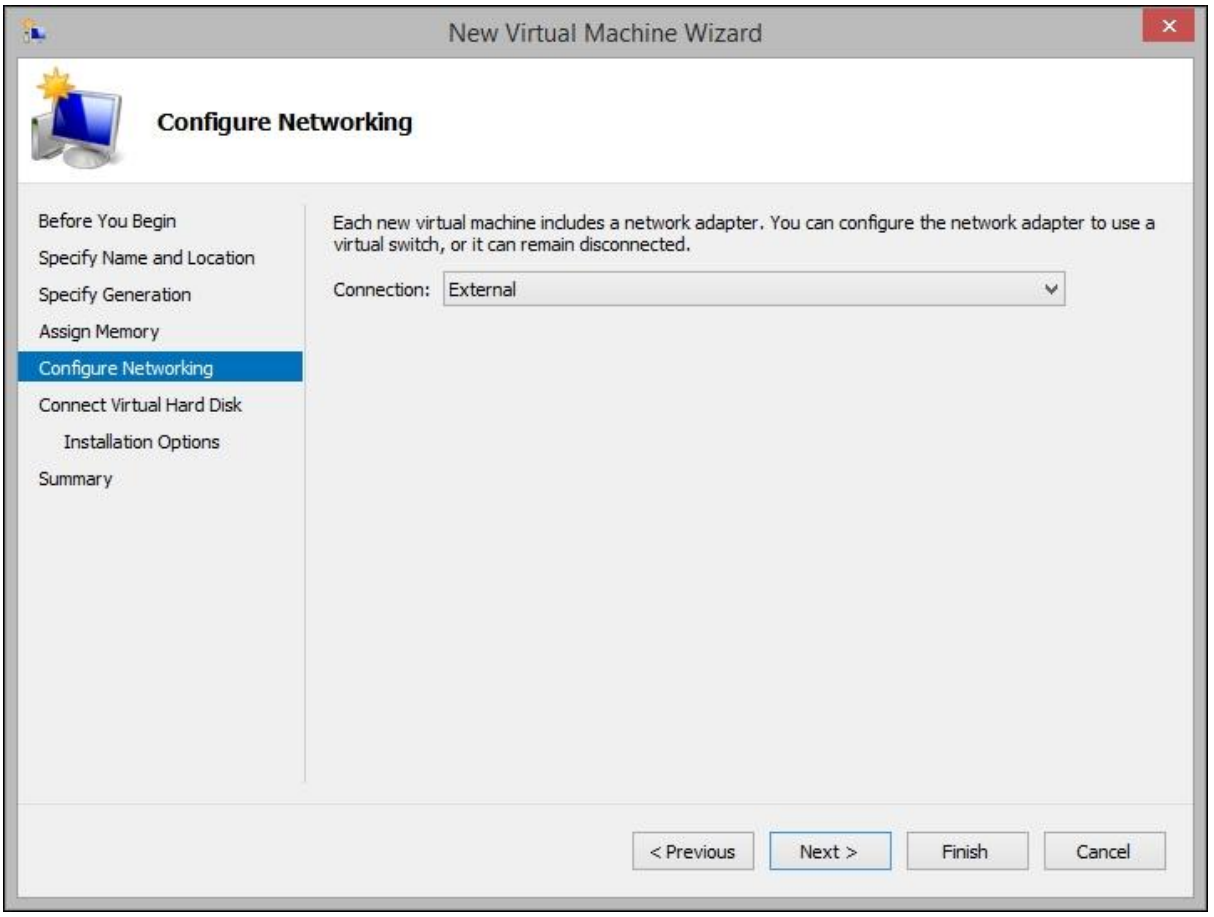
3. In our next selection, we'll specify **2,048** MB for **Startup memory**, and not check the box for **Use Dynamic Memory for this virtual machine**. We don't want the hypervisor to try to reduce the RAM footprint of our system while it's installing software, Windows Updates, and so on:



No Dynamic Memory here!

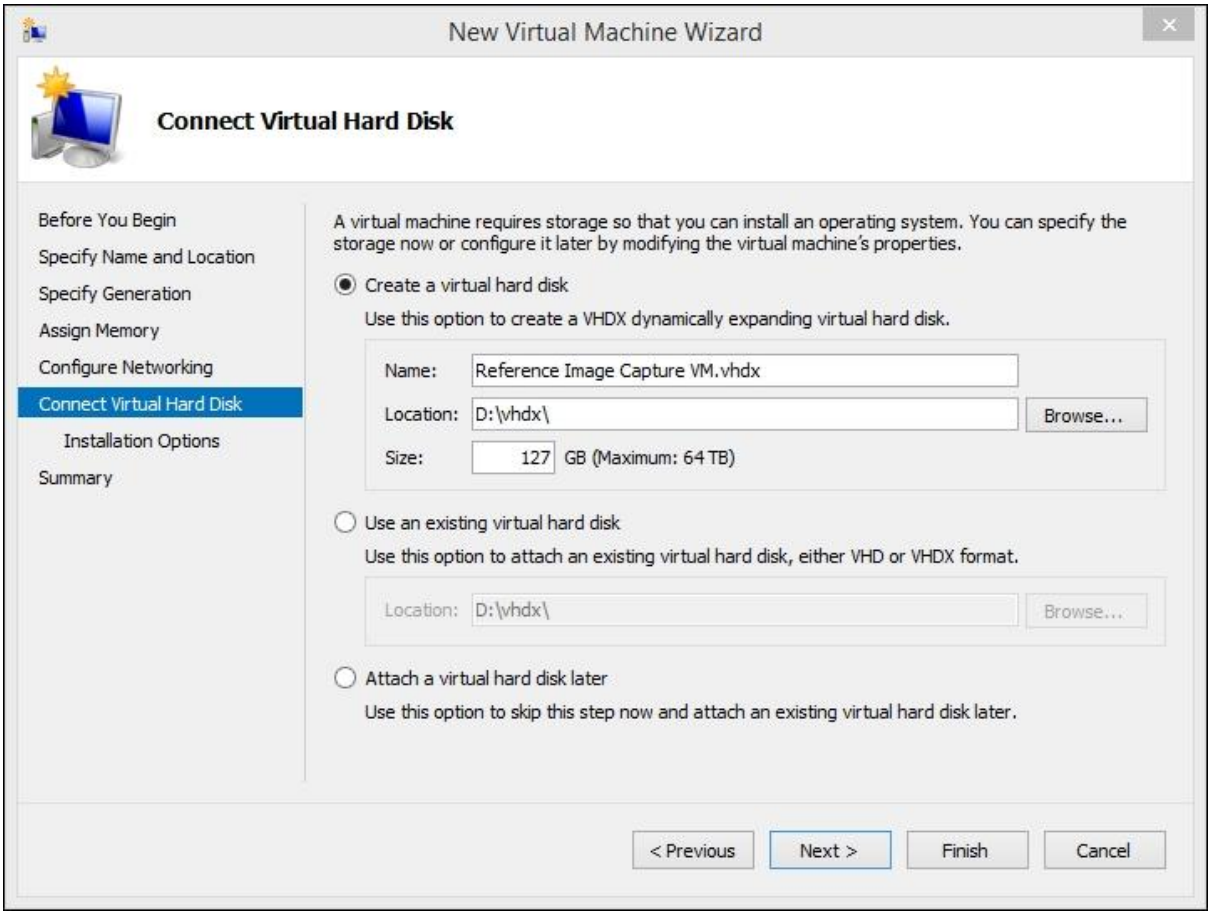
4. Then on the next screen, we need the virtual machine to be connected to a network that has access to several items. The deployment server, preferably a DHCP server, DNS, and either an external path to Windows Updates or an internal WSUS server. We will simply place the virtual machine in this network in Hyper-V.

5. We'll cover static IP address considerations later in this chapter:



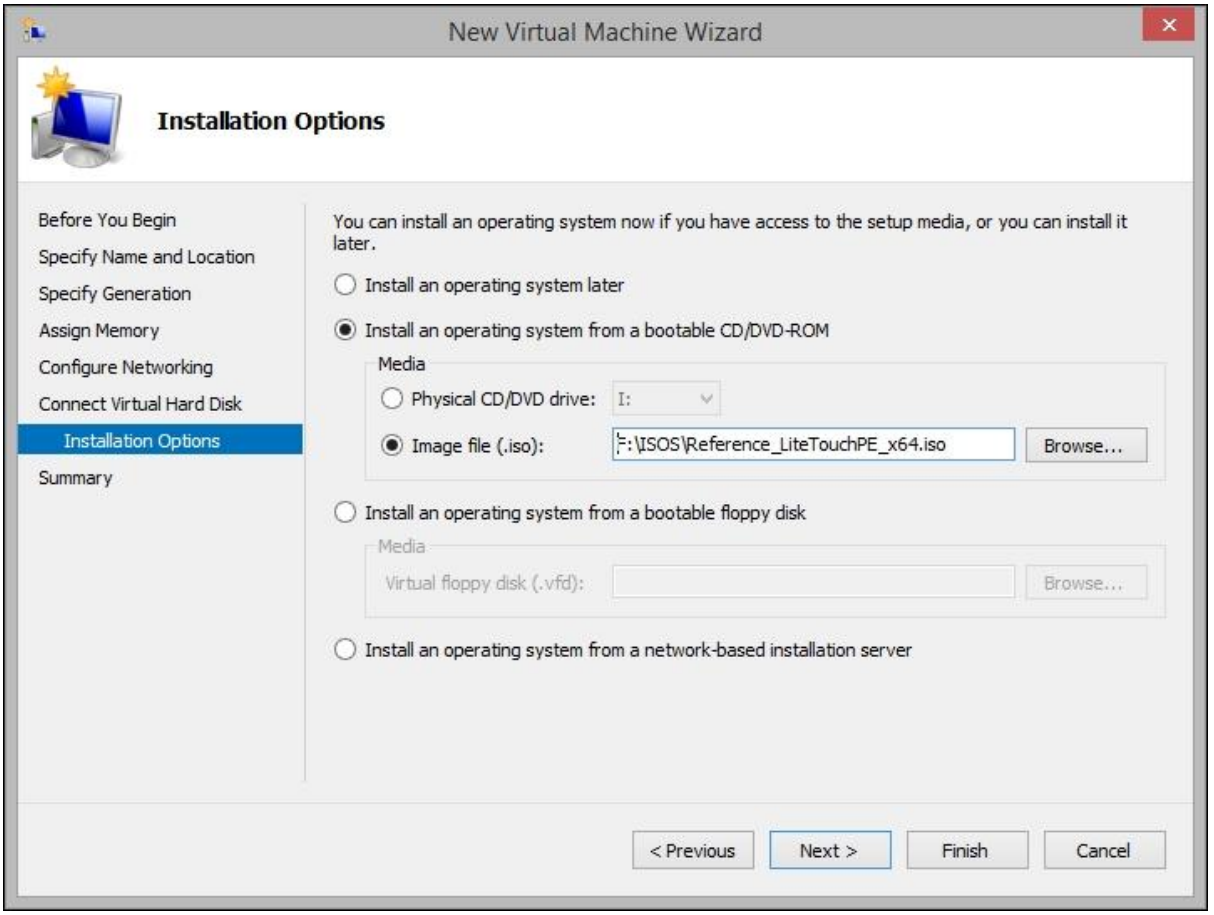
External network

6. On the next screen, we need to allocate a virtual disk. If we don't already have a virtual disk defined, we can simply ask Hyper-V to make one for us now. The drive location should be speedy and have space for a Windows installation and applications:



VHDX location

7. On the next screen we are asked whether we want to install an OS now? Well, sort of. What we want to do here is specify the ISO of the MDT reference share. This ISO will be configured in the `Bootstrap.ini` and `CustomSettings.ini` to know to talk to our reference share, run task sequences, and so forth. This ISO is located in `C:\ReferenceShare\Boot` on our MDT server. Copying it from the reference share to the Hyper-V host so that it appears in a local disk is recommended:

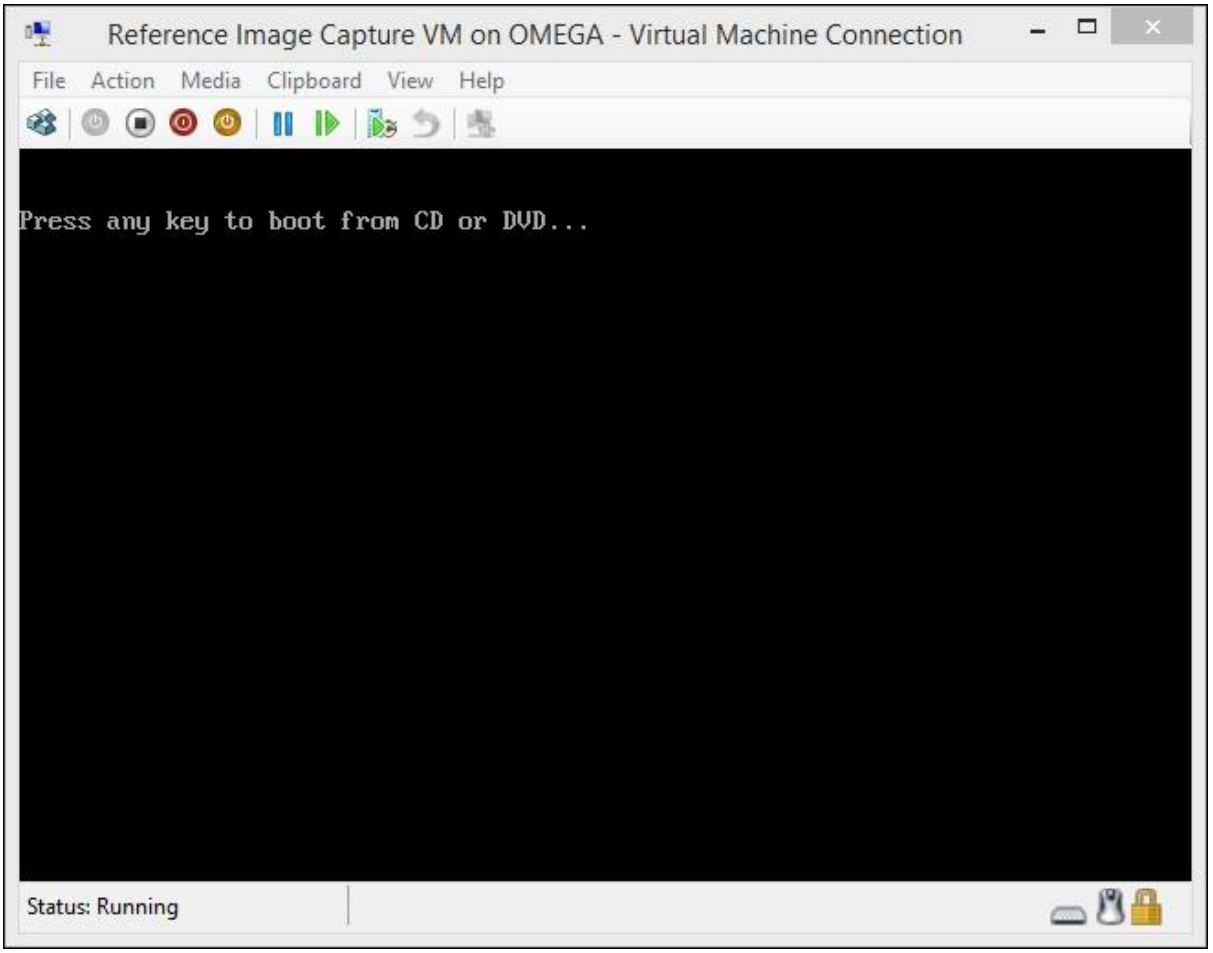


The ISO must be on a local volume for the Hyper-V host

8. Then, it is a simple matter of clicking **Finish** and starting the virtual machine. The machine will go through a boot process.

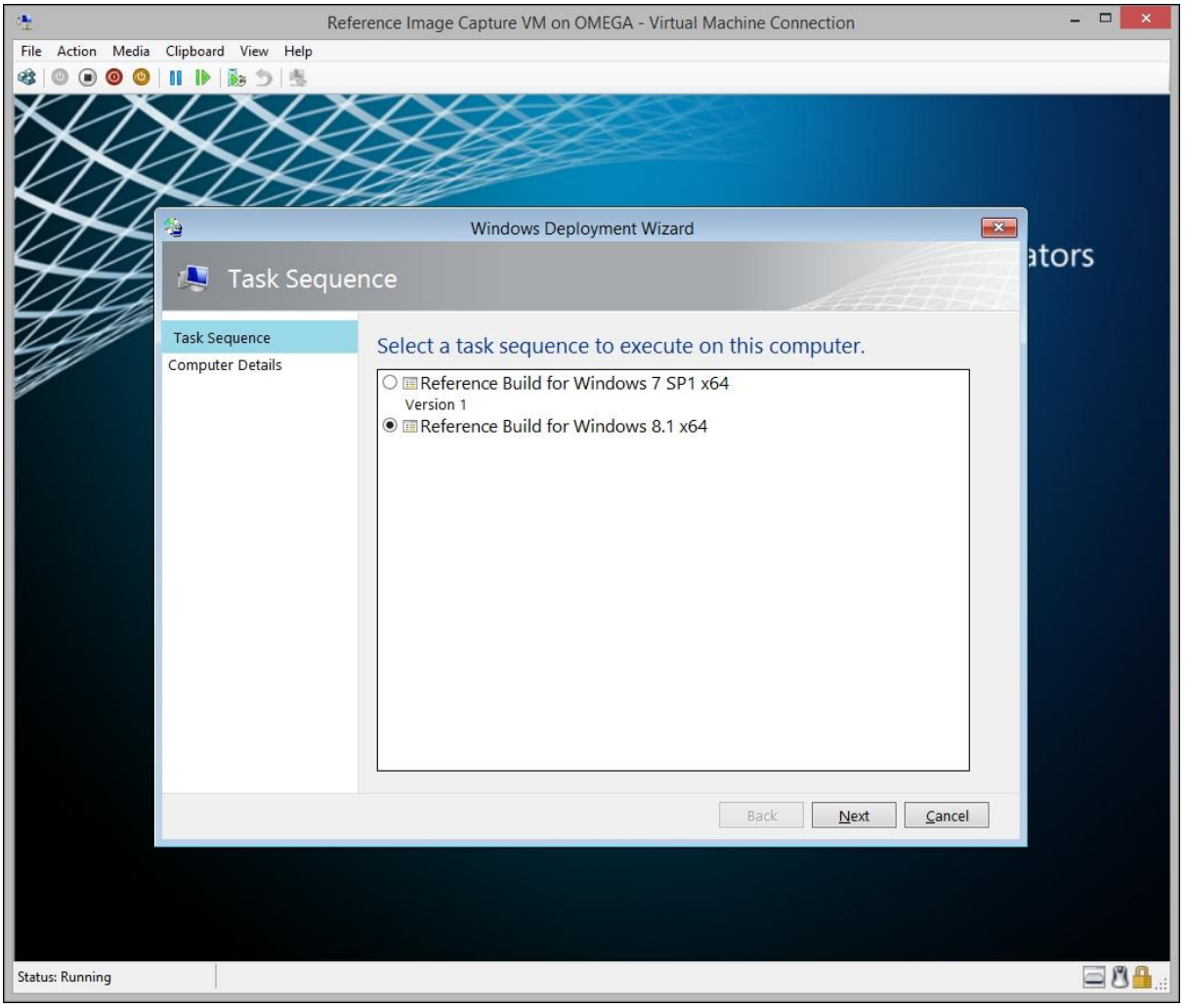
Deployment

The virtual machine will go through the boot process, as shown in the following screenshot:



Here, BIOS in the virtual machine has posted and it is booting up

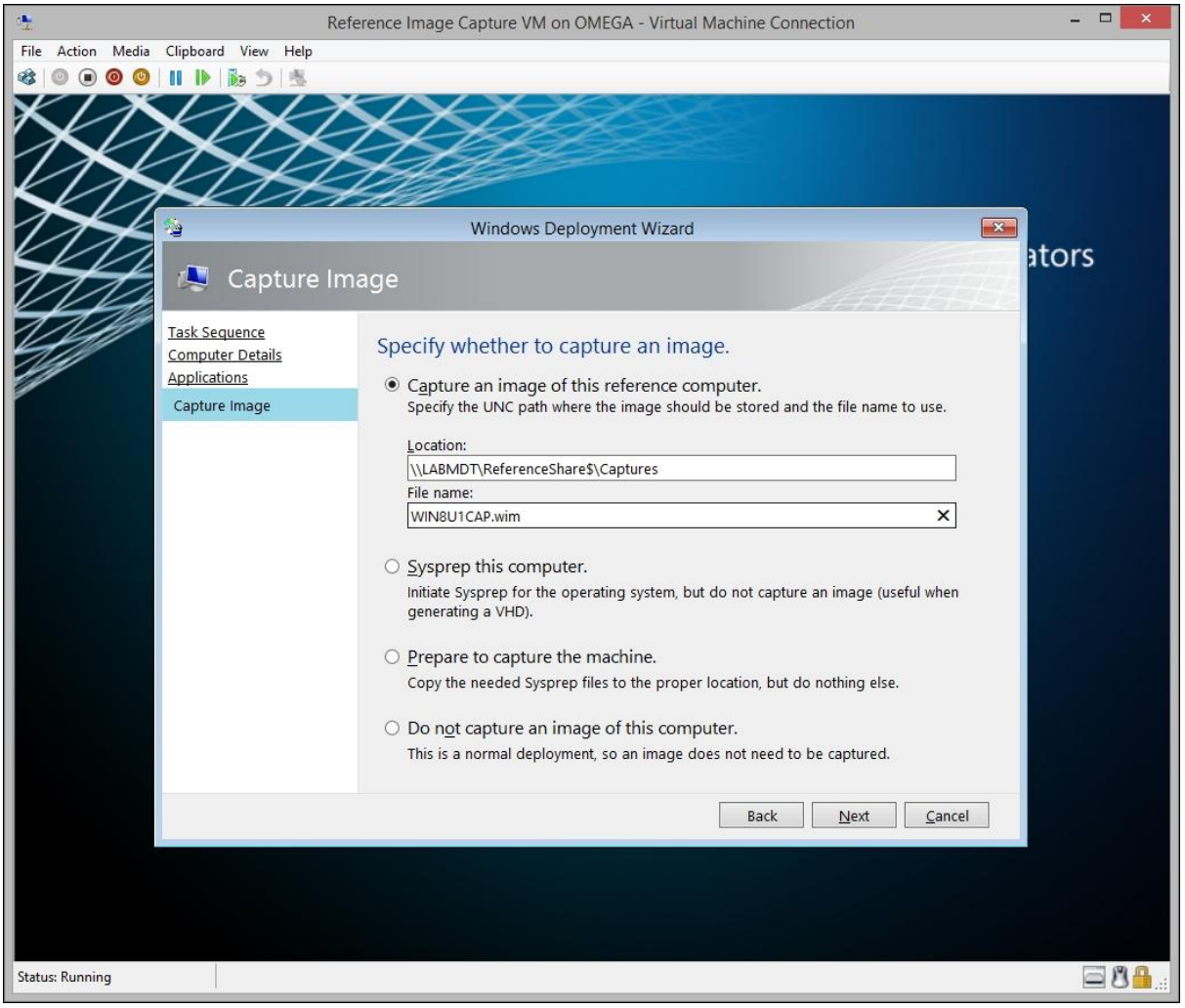
We must then specify the task sequence that we wish to run:



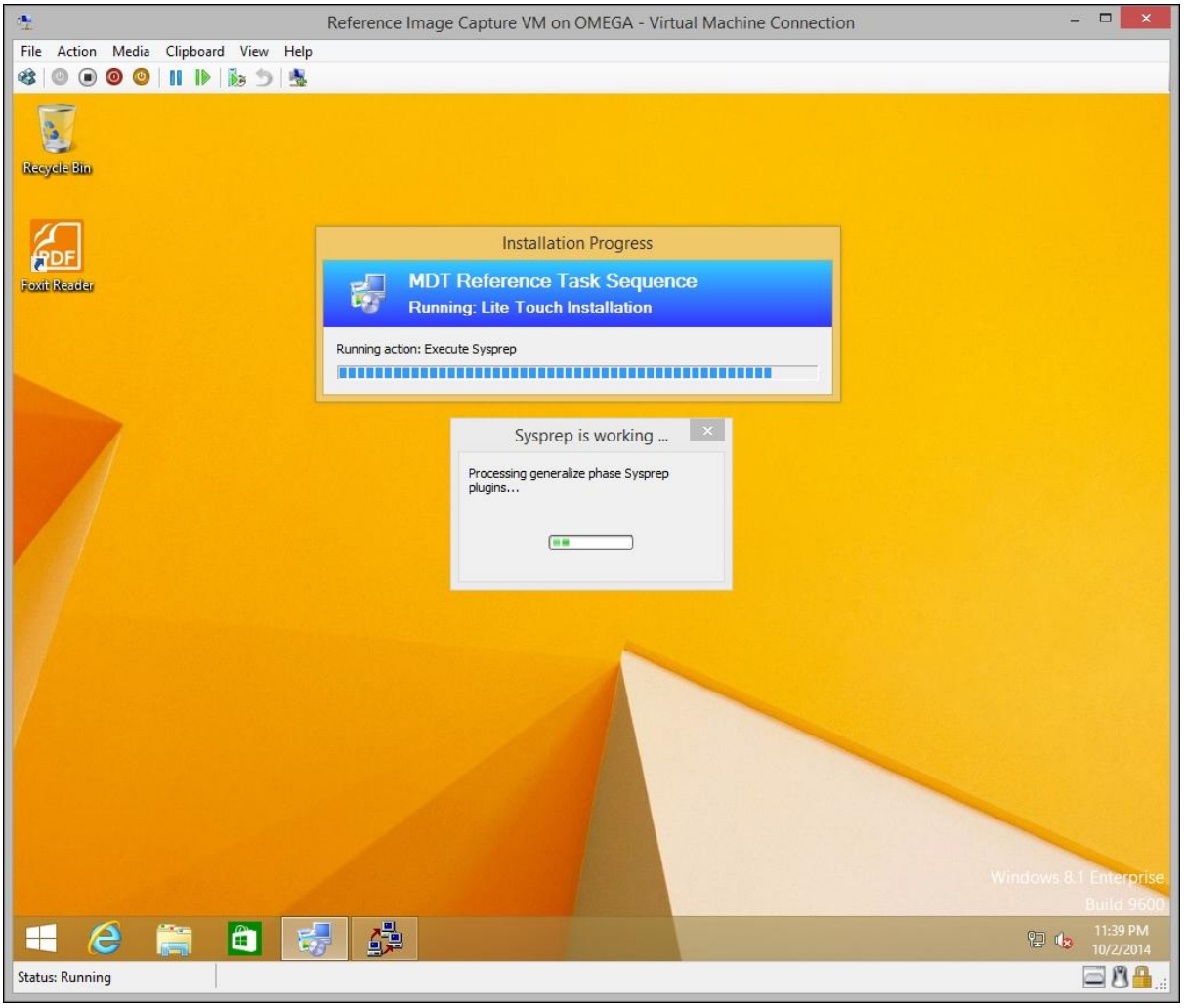
Note that now our task sequence engine skipped some screens. Our `CustomSettings.ini` has specified the following:

```
SkipCapture=NO
SkipAdminPassword=YES
SkipProductKey=YES
SkipBitLocker=YES
SkipDomainMembership=YES
JoinWorkgroup=Workgroup
SkipFinalSummary=YES
SkipLocaleSelection=YES
SkipSummary=YES
SkipTimeZone=YES
SkipUserData=YES
```

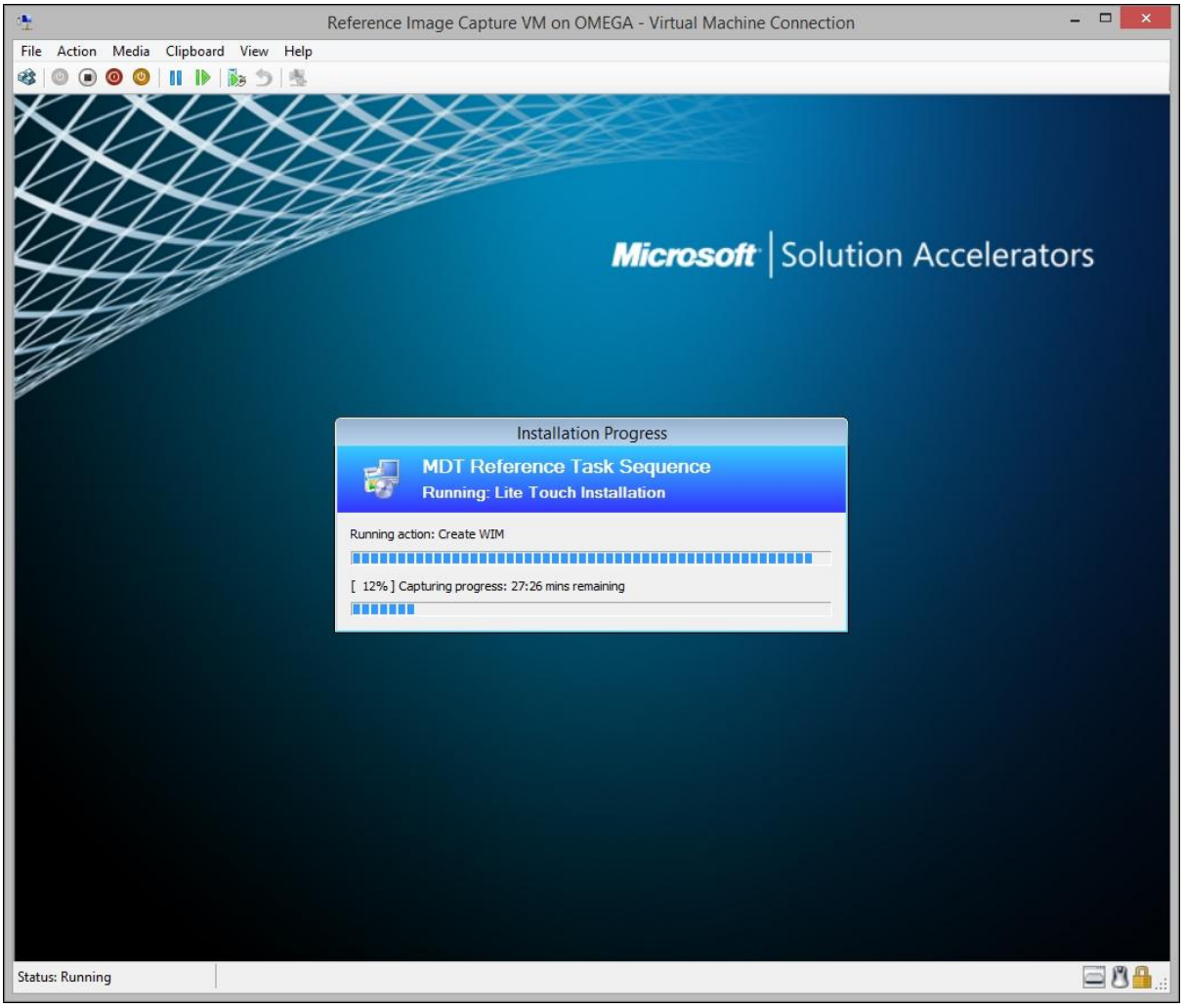
As we specified `SkipCapture=NO` in the preceding code, the **Capture Image** screen wizard is displayed as follows:



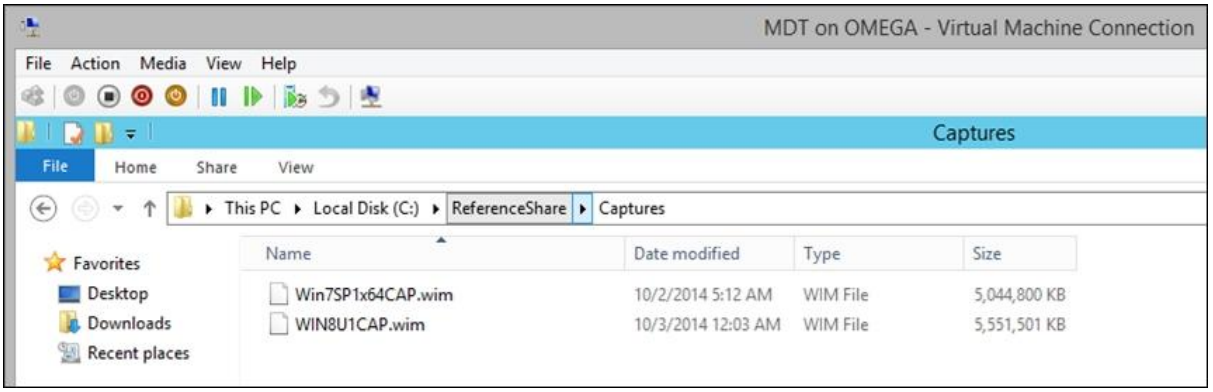
The task sequence will then execute through the OS install, apply Windows updates and application installations, and then Sysprep and capture the image:



After the Sysprep is complete, the WIM is captured. This cannot be done with the running system, so MDT needs to start a Win PE by applying Win PE to the virtual machines' disk, changing the boot entry to Win PE, then rebooting to Win PE, and running a DISM `/captureimage` command:



The end product after running both our task sequences should be that our `Captures` directory on the MDT reference share looks something similar to the following image:



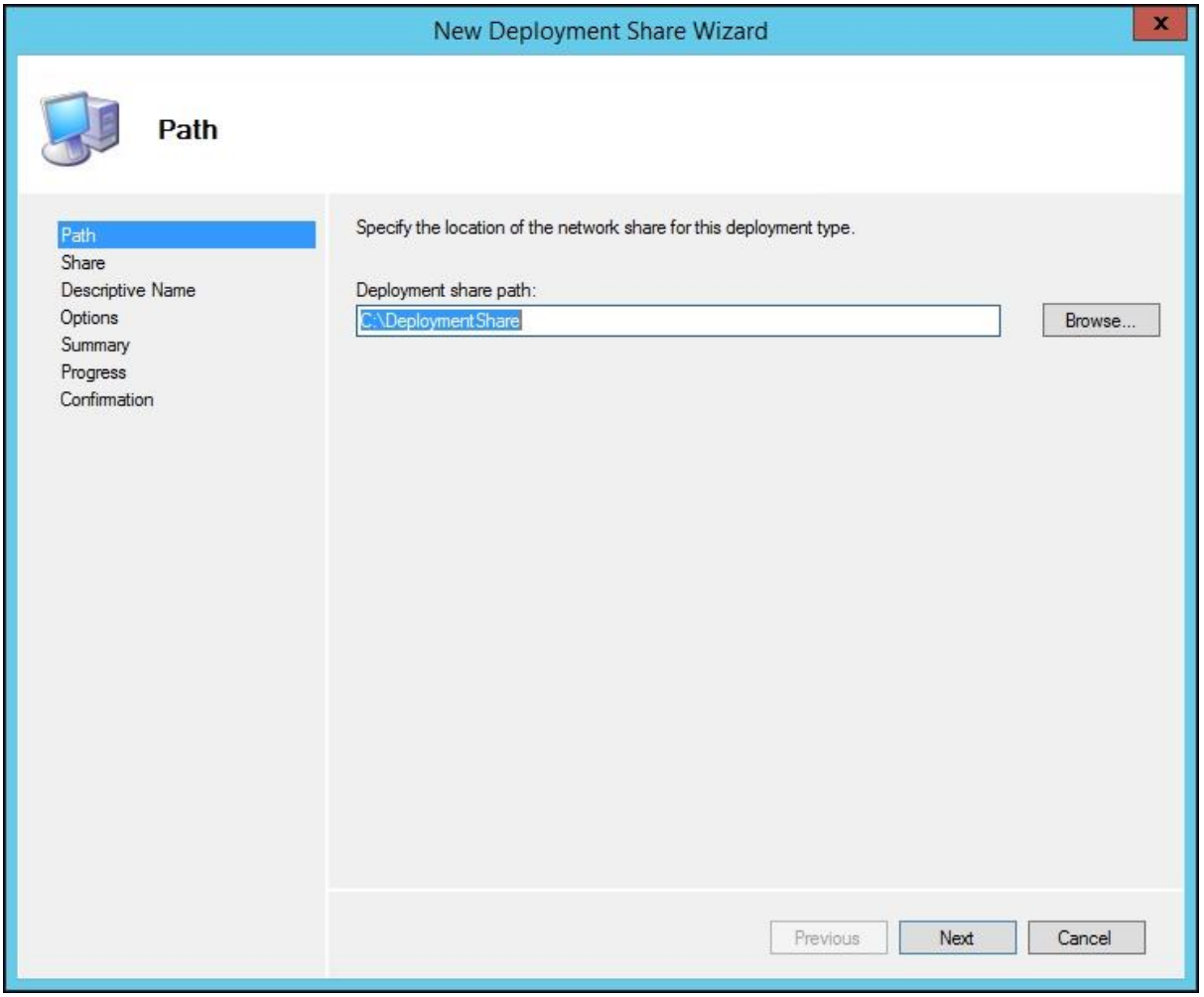
Here we can see that we have two WIM files, both approximately 5 GB in size.

Now the process is to import these WIM files as OSes into our deployment share. We haven't created a deployment share yet, but the steps are essentially the same as creating the reference share.

Deployment share

The deployment share is quite similar to our reference share. Most of the content of the `CustomSettings.ini` and `Bootstrap.ini` will be the same. The OSes of the deployment share are simply the WIM files, which we just captured, the product of our reference share task sequences. The applications will be complex drivers for specific hardware devices or applications applied post-OS deployment. However, our base images are somewhat set in stone at this point. The WIM files from our reference share form the base operating system of our deployment task sequences.

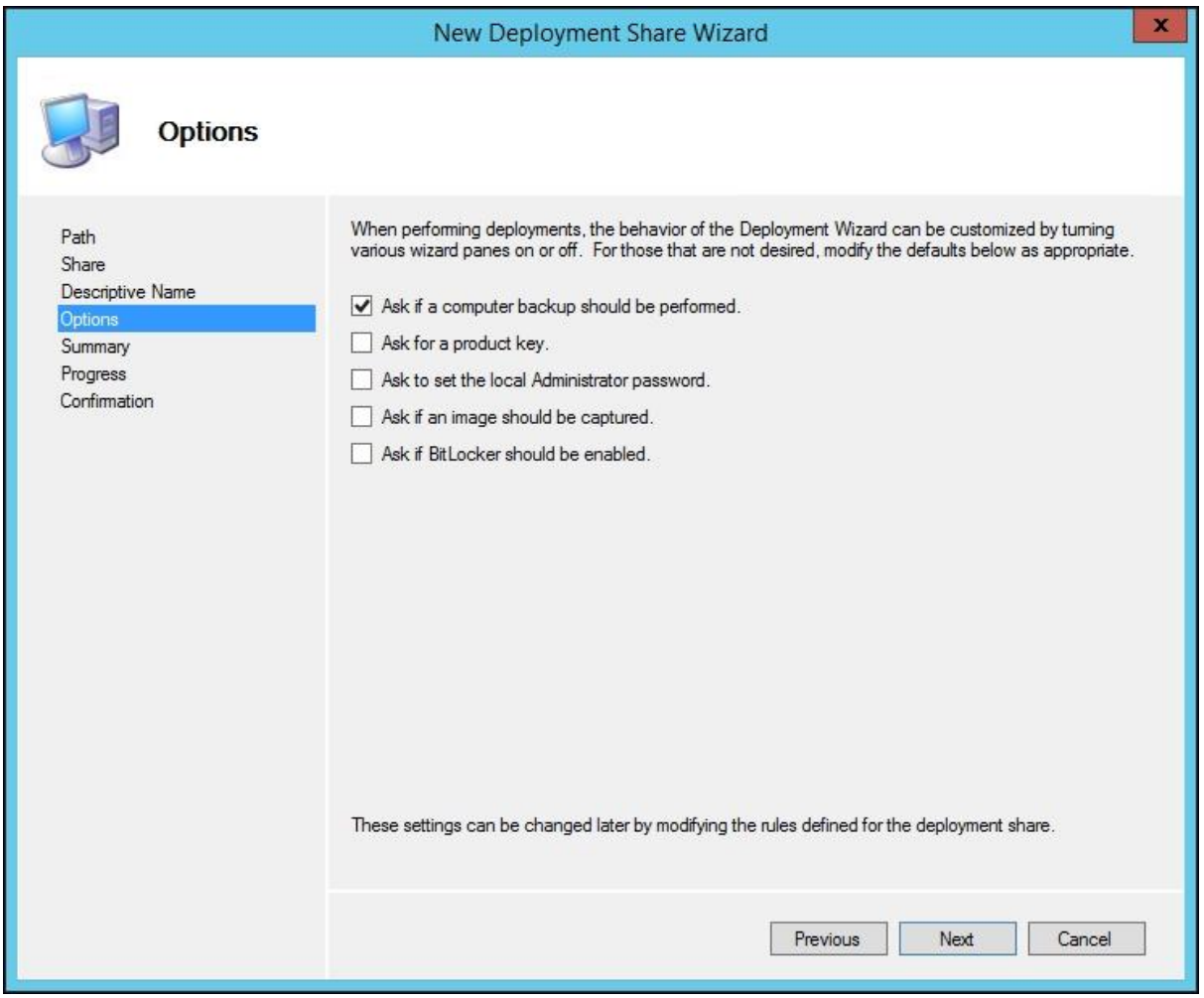
Again, creation of the deployment share follows the reference share, with some naming differences:



Name the directory `DeploymentShare` to keep our naming standards correct. We absolutely want to keep the reference share and deployment share work separate. This gives us the flexibility of experimenting in the reference share space, building baseline images, tweaking task sequences, and so on, without impacting

deployment share, where deployments will actually occur, replication with partners will take place, and so forth.

We will continue through the wizard until we get to the **Options** area, where depending on need, we'd likely want it set as shown in the following image:



We may need to backup a machine if we run the `litetouch.vbs` script on an existing Windows installation and perform a replace scenario. We don't want to capture an image though, so we uncheck this. We've already done our capture work in reference share.

Some changes that we might want to make in `CustomSettings.ini` are as follows:

```
SMSTSOrgName=MDT Deployment Task Sequence
SkipDomainMembership=YES
JoinDomain=Contoso
DomainAdmin=Administrator
```

So, it will look something similar to the following:

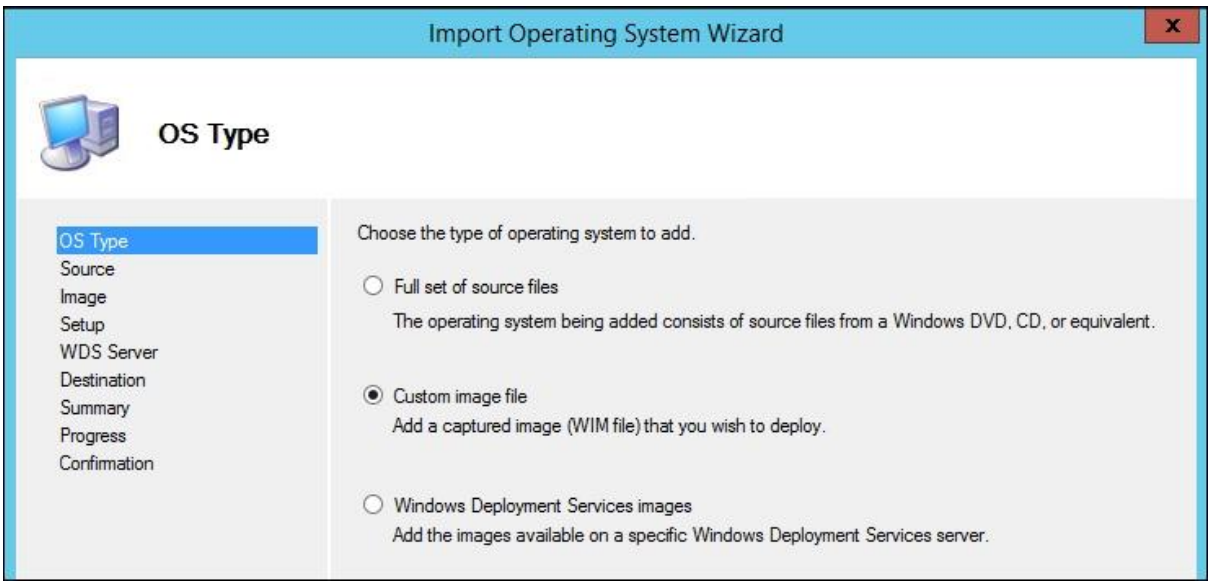
```
OSInstall=Y
SkipAppsOnUpgrade=YES
SkipCapture=YES
SkipAdminPassword=YES
SkipProductKey=YES
_SMSTSOrgName=MDT Deployment Task Sequence
SkipBitLocker=YES
SkipDomainMembership=YES
JoinWorkgroup=Workgroup
SkipFinalSummary=YES
SkipLocaleSelection=YES
SkipSummary=YES
SkipTimeZone=YES
SkipUserData=YES
TimeZoneName=Eastern Standard Time
UserID=administrator
UserDomain=contoso
UserPassword=Password
FinishAction=RESTART
```

TIP

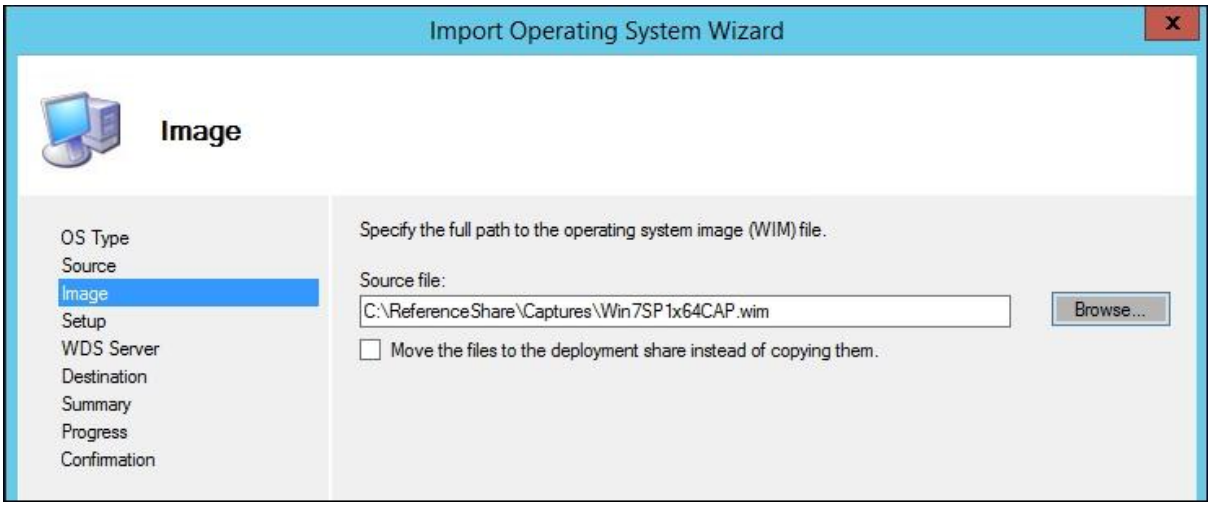
`FinishAction=RESTART` or `SHUTDOWN` is important in managed environments. If this is not present, the action will be to leave the freshly deployed machine that is joined to the domain and logged on as local administrator, which could leave the system vulnerable to end user shenanigans.

Don't forget to import Hyper-V driver additions, so WIN PE has them and update the deployment share as well. For Hyper-V on Server 2008 R2 or 2012 R2, log on to a virtual machine and insert integration service setup disk. If you do not have a virtual machine up and running, you can mount `%windir%\system32\vmguest.iso`.

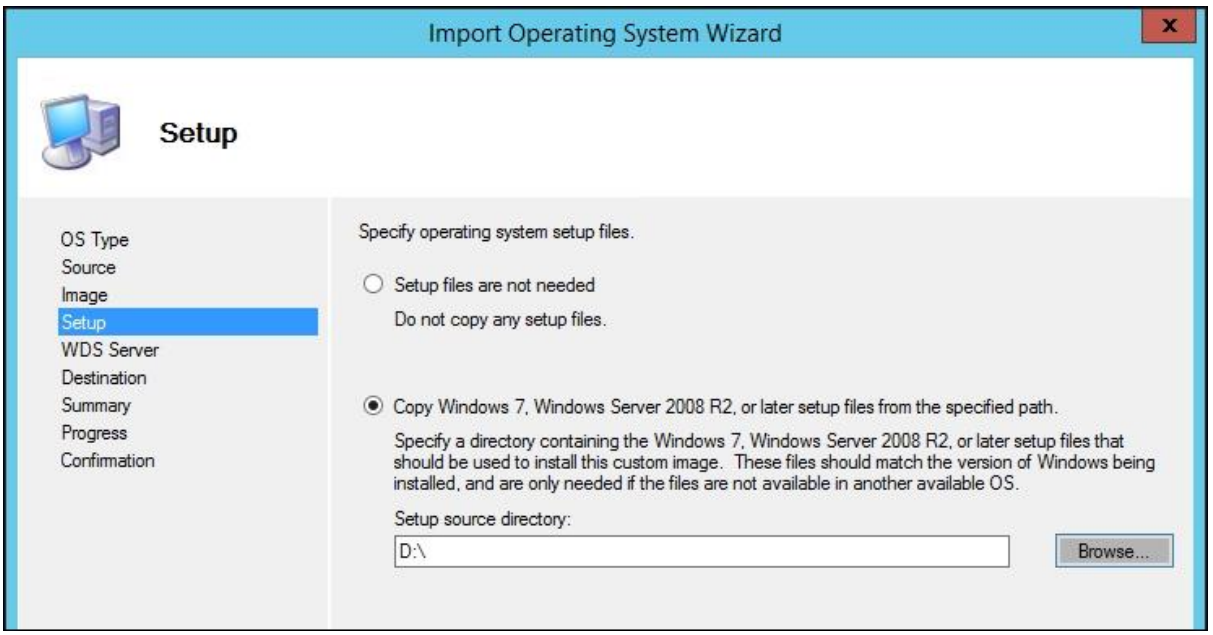
Here, we will import the custom WIM file as an OS:



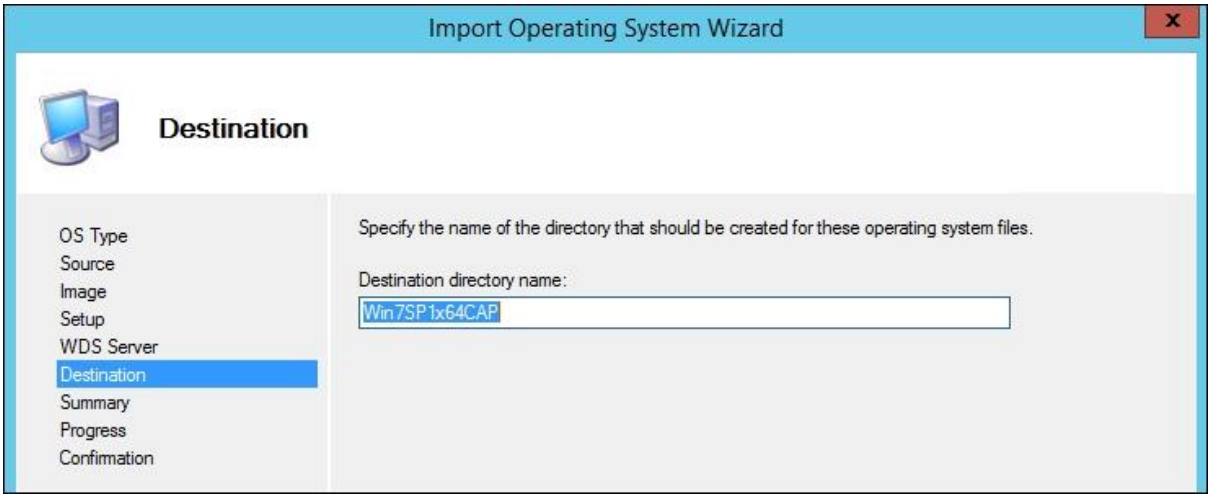
Also specify the file noted in the reference share. Note the checkbox, you can move the file to speed the process. However, I prefer to keep the WIM files in the reference share to keep a library of work:



After we specify the WIM, on the next screen, we will specify the setup files needed from the OS media. It is important to pick the one appropriate to bitness and version matching the WIM file. Point to the official ISO for the OS you built/captured the WIM from:



Then we give it a friendly name:



Deployment scenarios and network considerations

During the capture process, the task sequence follows the steps, determining the variables, making decisions whether to continue on error, and so on, all according to the task sequence created in the share. We've gone over the task sequence engine, settings, and variables, but we have not yet discussed the scenarios on the share itself, deployment strategies, replication for enterprise environments, and so on. We'll cover some of these scenarios now.

Firstly, deployment in most enterprises is considered a dangerous event. I've seen several misconfigured environments that started running SMS or MDT-based task sequences on production systems. User data is lost, days of productivity are lost, and IT careers are altered in a negative way. One such incident that is helpful to dissect occurred at a healthcare provider in the United States.

SMS advertised a deployment task sequence over an agent that told the agent to format the existing hardware and install Windows 7. This includes all laptops, desktops, and servers in the environment.

Deployment networks

For this reason alone, I typically advise customers to create a virtual network (VLAN) for deployments. The MDT deployment share resides on this VLAN, and hosts that are to be migrated are moved onto the VLAN temporarily for servicing. As laying down a fresh image on a host happens rarely (one would hope), this should mitigate the accidental task sequence push to all clients.

So, deploy in a VLAN. This brings us to the concept of how to configure this network.

Configuration of the deployment network

Use Multi-Cast on the deployment VLAN. Multi-Cast transmission will allow Win PE boot media to start listening to the broadcast anywhere in the image stream, as it is transmitted to all endpoints on the VLAN. Therefore, Win PE can start listening as soon as it is network-ready, and when the whole deployment transmissions is complete, it can begin running the task sequence.

DHCP should be configured on the network, even in an environment that does not use DHCP. The pain associated with manual static IP address assignments in a mass-deployment scenario is possible with a lot of work, or a DHCP server.

The network should be as fast as possible, given that it's going to be deploying and (potentially) migrating user data via **User State Migration Tool** (**USMT**). A sizing estimation for data migration should also be done. The size of the user profile area on your hosts * number of hosts being migrated at once = the general amount of storage you might need (multiply the result by two or three so that you have extra. You can never have enough storage).

More details about USMT and it's configuration will be discussed in Chapter 8 , *USMT - The User State Migration Tool*.

Geographical considerations

When we are deploying across geographical sites, we would want to use linked deployment shares (available in the advanced area of the MDT share). The key here is to manage your share centrally from the master deployment share, then use linked deployment shares to essentially act as a floodgate to your downstream deployment points (ideally hosted in DFS).

You may not (at the time of writing) utilize an active directory associated DFS share for replication. Therefore, a deployment standalone DFS infrastructure is recommended. This sounds like a lot of work; but in reality, it is quite simple to stand up.

So again, Master deployment share is linked to another share (which acts as a kind of floodgate). This link will reside as a folder structure in a standalone DFS configuration. You can then use block-level differential copying native in DFS to save bandwidth and time in replicating changes between geographical sites.

NOTE

Windows 10 All the concepts shown in this chapter are valid for Windows 10, but pay attention to the following points:

- Windows 10 as a guest virtual machine is only supported on Hyper-V on Server 2012 R2 or newer or on Hyper-V on Windows 10.
- I recommend using a Generation 2 virtual machine to create the Windows 10 images.
- If you plan to create a Windows 10 image used for inplace upgrade, only pure OS features on demand and patches are allowed. Do not add any application to an inplace upgrade Image.
- For a normal *wipe and reload* Windows 10 image, you can add applications as shown previously.